

AD-A179 777

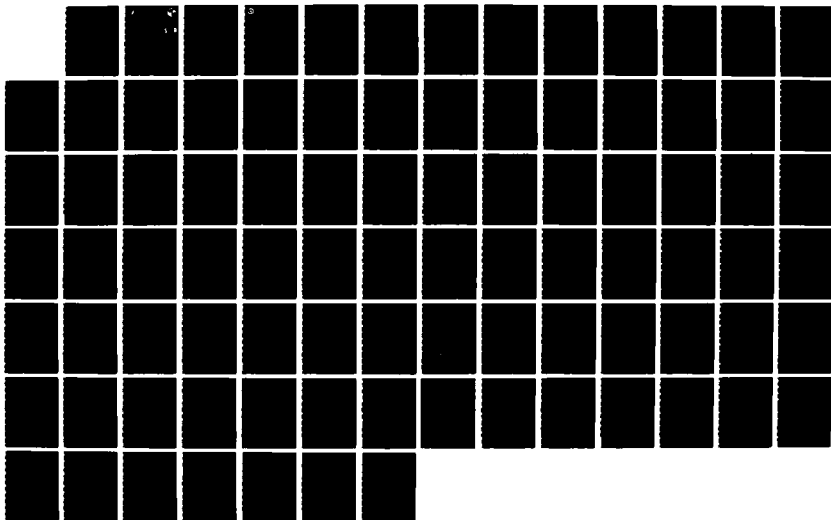
COMPUTER HARDWARE/SOFTWARE MONITOR(U) ARMY ELECTRONIC
PROVING GROUND FORT HUACHUCA AZ K E VAN KARSEM SEP 86

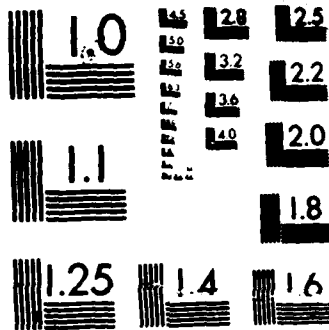
1/1

UNCLASSIFIED

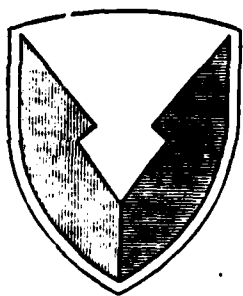
F/G 12/8

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



AD-A179 777

DTIC FILE COPY



AD NUMBER _____

TECOM PROJECT NO. 7-CO-R86-EPO-005

METHODOLOGY INVESTIGATION

FINAL REPORT

COMPUTER HARDWARE/SOFTWARE MONITOR

BY

K. E. VAN KARSEN

Software and Automation Division
Electronic Surveillance and Security Test Directorate

US ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, AZ 85613-7110

MARCH 1987

Prepared for:

US Army Test & Evaluation Command
Aberdeen Proving Ground, MD 21005-5055

Approved for Public Release;
distribution unlimited.

DTIC
ELECTE
APR 29 1987
S D

87 4 28 007

DISPOSITION INSTRUCTIONS

Destroy this report in accordance with appropriate regulations when no longer needed. Do not return it to the originator.

DISCLAIMER

Information and data contained in this document are based on input available at the time of preparation. Because the results may be subject to change, this document should not be construed to represent the official position of the U.S. Army Materiel Command unless so stated.

The use of trade names in this report does not constitute an official indorsement or approval of the use of such commercial hardware or software. This report may not be cited for purposes of advertisement.

VAX and VMS are trademarks of Digital Equipment Corporation.



DEPARTMENT OF THE ARMY
U.S. ARMY ELECTRONIC PROVING GROUND
FORT HUACHUCA, ARIZONA 85613-7110

REPLY TO
ATTENTION OF

STEEP-MT-DA

NOV 13 1986

SUBJECT: Methodology Investigation Final Report, TECOM Project No.
7-CO-R86-EPO-005

Commander
US Army Test and Evaluation Command
ATTN: AMSTE-AD-M
Aberdeen Proving Ground, Maryland 21005-5055

1. Reference RDTE Methodology Improvement Program Directive, Computer Hardware/Software Monitor, TECOM Project No. 7-CO-R86-EPO-005, 15 Oct 85.
2. The subject final report for the Computer Hardware/Software Monitor is enclosed per reference.
3. Point of contact at this activity is Mr. Edward L. Anderson, AV 879-1870/1879.
4. USAEPG - Providing leaders the decisive edge.

FOR THE COMMANDER:

G. W. DURBIN
Chief, Electronic Surveillance
and Security Test Division

Encl



Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

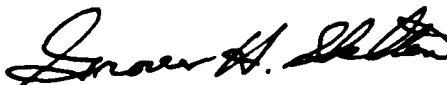
AMSTE-TC-M (STEEP-MT-DA/13 Nov 86) 1st End Mr. Knox/brt/AV 298-3677
SUBJECT: Methodology Investigation Final Report, TECOM Project No.
7-CO-R86-EP0-005

Cdr, U.S. Army Test and Evaluation Command, Aberdeen Proving Ground, MD
21005-5055 12 FEB 1987

TO: Commander, U.S. Army Electronic Proving Ground, ATTN: STEEP-MT-DA,
Fort Huachuca, AZ 85613-7110

1. Subject report is approved.
2. TECOM - Providing Soldiers the Decisive Edge.

FOR THE COMMANDER:



Encl wd

GROVER H. SHELTON
Chief, Methodology Improvement
Division
Directorate for Technology

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TRMS No. 7-CO-R86-EPO-005	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) METHODOLOGY INVESTIGATION FINAL REPORT COMPUTER HARDWARE/SOFTWARE MONITOR		5. TYPE OF REPORT & PERIOD COVERED Methodology Investigation Final Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) K. E. Van Karsen		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Electronic Proving Ground Fort Huachuca, AZ 85613-7110		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1W665702D625
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Test and Evaluation Command Attn: AMSTE-TC-M Aberdeen Proving Ground, MD 21005-5055		12. REPORT DATE September 1986
		13. NUMBER OF PAGES 82
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Software Measurement Hardware Monitor Software Monitor <i>computer controlled</i> <i>software controlled</i>		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Previous investigations established methods for evaluating software or systems containing multiple interactive processors. These investigations resulted in the development of a hybrid monitor concept: the integration of both hardware and software monitor capabilities into a central monitor system. This investigation resulted in revision of the specification documents and continued design and implementation of the prototype hybrid monitor. The prototype was completed sufficiently to validate the HM concept and demonstrate the feasibility of collecting various measures of performance.		

(BLANK PAGE)

TABLE OF CONTENTS

FOREWORD	v
----------------	---

SECTION 1. SUMMARY

<u>Paragraph Number</u>		<u>Page</u>
1.1	Background	1
1.2	Objective	1
1.3	Summary of Procedures	2
1.4	Summary of Results	2
1.5	Analysis	3
1.6	Conclusions	3
1.7	Recommendations	4

SECTION 2. DETAILS OF INVESTIGATION

2.1	Background and Initial Efforts	5
2.1.1	Technology Review	5
2.1.2	Target System/MOP Selection	6
2.2	Prototype HM Development	6
2.2.1	Hybrid Monitor Components	6
2.2.2	Prototype Hybrid Monitor Configuration	9
2.3	MOP Collection	9

SECTION 3. APPENDIXES

Appendix

A.	Methodology Investigation Proposal	15
B.	References	21
C.	Acronyms and Abbreviations	25
D.	Prototype Hardware Monitor Characteristics	29
E.	Measures of Performance	33
F.	Hybrid Monitor Methodology	43
G.	Selected MOP Results	55
H.	Distribution	77

(BLANK PAGE)

LIST OF FIGURES

<u>Figure</u>		<u>Page</u>
1	HM Functional Block Diagram	8
2	Prototype HM Functional Diagram	10
3	Performance Analysis Methodology	46
4	Software Monitor Configuration	48
5	Hardware Monitor Configuration	49
6	Hybrid Monitor Configuration	51

LIST OF TABLES

<u>Table</u>		<u>Page</u>
I	HM Target System Selection Criteria	7
II	MOP Test Configurations	11
III	Prototype MOP Categories	13
IV	Measures of Performance	36
V	Comparison of Monitor Characteristics	52

(BLANK PAGE)

FOREWORD

Ultrasystems Defense and Space Systems, Incorporated,
Sierra Vista, Arizona assisted in the preparation of this
document under Contract Number DAEA18-83-C-0003.

(BLANK PAGE)

1. SUMMARY

1.1 Background

The increasing complexity of Command, Control, Communications and Intelligence (C³I) has dictated the need to use multiple processors to enhance system reliability and to provide timely processing in systems that process large volumes of data. The trend is toward distributive processing systems which will have multiple interactive processors and common software functions. Very little has been done to establish comprehensive test procedures for evaluating these systems, particularly under full load conditions.

Previous investigations into a test methodology for software testing of multiprocessor systems identified various monitoring techniques applicable to performance testing [see references 1 and 2, appendix B]. Although each approach provided certain advantages for the particular application, none provided a general purpose, flexible monitoring capability suited for a multiprocessor environment. Hardware monitors provide a nonintrusive capability suitable for microprocessor based systems. Communication link monitors are important in the testing of C³I systems, but can only indirectly measure performance parameters internal to the target system. Software monitors are inherently flexible and can perform complex data collection, but are intrusive and require target system resources to operate.

Because of the limitations of using either software or hardware monitors alone, a concept for general purpose multiprocessor testing evolved. This concept, called the hybrid monitor (HM), involves the use of both hardware and software monitors in conjunction with a central controller. The central controller's purpose is to control the hardware and software monitors. This hybrid approach embodies the best features of both hardware and software monitors. The central controller is also able to correlate data from multiple sources, essential for multiprocessor applications.

The HM concept was analyzed further to identify possible test configurations and to determine the general applicability to multiprocessor system testing. Efforts were initiated to develop the concept into a prototype tool to be used for refining and validating the methods and procedures for multiprocessor software testing. These efforts included the selection of a hardware monitor and the top-level design and implementation of selected operator interface functions for the prototype HM.

Other efforts during these investigations included the development of proposed measures of performance (MOPs) suitable for multiprocessor systems. The proposed MOPs were further refined as the concept of the HM evolved.

The result of the earlier investigations was that a prototype HM was partially developed, but was not sufficiently complete to function in a hybrid configuration. This precluded validation of the HM method and MOPs for multiprocessor software testing.

1.2 Objective

The objective of this investigation was to validate the proposed methodology of a hardware/software hybrid monitor as a tool for testing systems

containing multiple interactive processors and to verify that a HM can collect the MOPs proposed during previous investigations.

1.3 Summary of Procedures

Initially, current technology in hardware/software monitoring was reviewed to determine whether advances in the technology had invalidated the conclusions and recommendations of previous investigations. The investigation proceeded with the selection of a target test system to validate the HM concept and verify the feasibility of collecting the proposed MOPs.

Hardware, software, and central controller functions were then enhanced to provide a HM capable of collecting the proposed MOPs. Examples of MOPs from various categories were then selected to demonstrate the ability to acquire performance measures with a mixture of hardware and software monitoring techniques.

1.4 Summary of Results

The review of the current technology in hardware/software monitoring confirmed the conclusions and recommendations made in previous investigations. Although recent technological advances did not invalidate previous recommendations to complete and validate a prototype HM, it was noted that considerable progress is being made in hardware monitor capabilities and in the speed, complexity, and packaging of components which potentially will appear in future test items.

A processor (T11) on a communication interface (KCT32) used on the Test Item Stimulator (TIS) test driver was chosen as a target system for hardware monitoring. Availability was the primary factor in this choice, although another consideration was that it represented more current technology compared to alternative candidate systems. Software processes executing on a VAX were selected for software monitoring because the target software could be executed on the same processor used to host the HM, eliminating the need for a separate computer system dedicated to monitoring efforts.

Hardware, software, and central controller functions of the HM were completed sufficiently to allow demonstration of a hybrid monitoring capability (simultaneous hardware and software monitoring). This effort included interfacing the hardware monitor (Tektronix DAS 9129) to the VAX via an IEEE-488 link, developing software kernels and data collector routines to acquire MOPs, and completing the necessary central controller functions.

The feasibility of collecting MOPs was verified by developing HM test configurations for selected MOPs. Eight test configurations with hardware, software, and hybrid monitoring were defined to measure twelve MOPs. Hardware monitor MOPs required that probe points be determined and a hardware monitor setup defined. Performance data was collected by running the HM, specifying the desired test configuration and storing the collected MOPs for post-test analysis.

All MOPs were successfully collected and various HM configurations demonstrated on single and multiprocessor target systems. The HM configurations included: software monitor on VAX, hardware monitor on T11, both

software and hardware monitors simultaneously, and one test with two software and one (only one DAS was available) hardware monitor.

Some limitations were observed with respect to collecting the various MOPs. The most significant limitations concerned the data collection bandwidth of the hardware monitor component of the HM. The buffer size limited the quantity of information collected during an acquisition period; delays in transferring data to the VAX prevented continuous measurement of some MOPs (off loading data and acquisition are not concurrent); and lack of on board time-of-day clock required that probes be dedicated to monitoring an external clock source. No significant limitations were experienced with the software monitors since potential problem areas, such as insufficient storage, were not a factor on the target system.

As designed, the prototype HM is capable of accommodating up to ten concurrent monitors of any combination of hardware and software types. Demonstration of the HM capabilities included operation of three concurrent monitors on two separate processors. HM host system resource requirements during collection of selected MOPs were typically less than ten percent of the available resources.

A and B-level specifications [see references 3 and 4, appendix B] for a full-scale HM tool have been generated. Operation of the prototype HM is described in the technical user's manual [see reference 5, appendix B].

1.5 Analysis

Recent advances in technology since previous investigations have not invalidated the concept of a HM. However, trends in hardware and hardware monitoring technology warrant periodic reviews. In particular, available hardware monitors should be examined before a selection is made for a full-scale HM. Improvements in monitoring techniques would require appropriate changes to the HM specifications.

The development and operation of the prototype HM in a hybrid mode on a multiprocessor test configuration demonstrates the validity of the HM concept. The feasibility of collecting the proposed MOPs was shown, however, given the limitations of the hardware monitor and HM host system, it is easy to conceive of MOPs which would exceed the HM performance capability. Use of more advanced monitoring equipment would not totally alleviate this situation, unless the performance characteristics of the host system and hardware monitor greatly exceeded the data collection bandwidth required by a given MOP on the target system.

1.6 Conclusions

This investigation successfully completed the development of a prototype HM, validated the concept in a multiprocessor test configuration, and demonstrated the feasibility of collecting various MOPs. Development of a full-scale HM appears feasible, however, specifications should be revised to reflect any significant advances in technology.

1.7 Recommendations

The following recommendations are made:

a. Monitoring of emerging methodologies for software testing of multi-processor systems should continue. In particular, new developments in hardware and software monitoring and MOPs should be examined with respect to the HM methodology. Refinements to the HM design and proposed MOPs may be expected from appropriate technological advances.

b. A methodology investigation should be established to develop a full-scale HM. The central controller function could reside in a Test Item Stimulator (TIS), a test driver developed at USAEPG. This proposed investigation would result in a methodology validated on C³I system tests, as well as refinement of the HM architecture and associated MOPs.

2. DETAILS OF INVESTIGATION

The computer hardware/software monitor investigation reviewed technical advances in monitoring methodologies, selected a target system and MOPs, completed a prototype HM configuration, and verified the feasibility of collecting various MOPs. Results of these efforts are summarized below and in the appendices. Details of the HM concept were supplied in the previously referenced methodology investigation reports, and in HM system documentation (available upon request).

2.1 Background and Initial Efforts

Previous investigations examined methods for testing systems containing multiple processor architectures. The concept of a hybrid monitor, combining both hardware and software monitor techniques, was developed into specifications for a prototype tool. Further effort resulted in a refinement of the requirements and a partially developed controller for the HM. The existing software resides in a VAX-11/VMS environment; the hardware monitor component consists of a Tektronix DAS 9129.

Prior to completing the development of the HM, the literature and efforts of other activities in hardware/software monitoring were examined briefly to verify the approach of this investigation. The effect of advances in hardware monitor and hardware technology was examined and a target system chosen for demonstrating the collection of MOPs with the prototype HM.

2.1.1 Technology Review

The hardware monitor (Tektronix DAS 9129 logic analyzer) possessed sufficient capability to continue development of the prototype hybrid monitor (see appendix D). As future hardware components operate at increasingly higher clock rates the data acquisition boards in the DAS may be upgraded to meet any foreseeable requirement (up to 2 GHz). However, the increasing logic density and new packaging (e.g., leadless chip carriers) of integrated circuits is expected to limit the signals accessible for monitoring.

Advances in hardware monitor design should reduce the cost and effort of adapting to new processor designs. Although the DAS requires fewer specialized "personality modules" than other logic analyzers, the trend is toward a product with generic adaptation capability. Another trend is the more sophisticated data reduction and analysis (DRA) processing performed by the logic analyzers. For some minimal performance monitoring applications a separate controller with DRA functions might be unnecessary.

While the capabilities of hardware monitors are improving, three factors require that a hybrid monitor configuration include a central controller (as in the proposed hybrid monitor architecture). The limited storage capability of present monitors requires a controller for logging and DRA of acquired data. Also, correlating events in a multiprocessor environment dictates that events be time-stamped and processed by a single controller. Finally, the need to correlate data from hardware and software sources requires that a hybrid monitor architecture include a central controller.

2.1.2 Target System/MOP Selection

Various target systems, including the Position Location and Reporting System (PLRS) and TIS, were considered for demonstrating the hybrid monitor concept. Availability was the critical factor in choosing a system (which precluded use of PLRS), followed by less important selection criteria (see table I). The final trade-off analysis compared Data General and KCT (T11 and 68000 processors) candidate systems. The KCT, an intelligent communications component of the TIS system, appeared to offer more advantages, and further effort focused on this device (specifically the T11 processor), as the target system for hardware monitoring. Software residing on the VAX was chosen for software monitoring because of the ready availability of software which interacted with the KCT and the availability of software monitor functions provided by operating system services.

MOPs for the target systems (T11 for hardware monitoring, VAX for software, and both for hybrid) were selected from those proposed in previous investigations (see appendix E). These are discussed further under MOP collection.

2.2 Prototype HM Development

The work remaining to demonstrate a prototype HM included completing the hardware and software monitor portions, developing the necessary central controller functions, writing software kernels to collect specific MOPs, specifying probe points and setups for the hardware monitor, and interfacing the DAS (hardware monitor) to the HM host (VAX). Completion of the prototype HM followed the HM methodology architecture established in prior investigations (see appendix F).

2.2.1 Hybrid Monitor Components

A multiprocessor HM is considered to consist of three major components (see figure 1):

a. Central controller. This is a computer system which serves as the nucleus for monitoring the system under test (SUT or target system). Both the hardware and software monitor components report the data which they are collecting to the central controller for:

- (1) Collection/Correlation.
- (2) Storage/Retrieval.
- (3) Real-Time/Post-Analysis.
- (4) Display.

b. Software monitor/kernels. Software monitors are data collection/reporting functions which extract SUT performance information using software kernels. These kernels, analogous to the probes of a hardware monitor, are embedded within the SUT software.

c. Hardware monitors. These are devices which extract SUT performance indicators using probes, comparators, and counters to monitor signals. These

Table I. HM Target System Selection Criteria

The following criteria were used as a basis for the selection of a target system for the hybrid monitor.

1. Migration of techniques to future systems. For the methodology to be of practical use, the HM concept should be demonstrated on a system employing current technology.
2. Multiple processor configuration. The target system should provide a multiprocessor environment to demonstrate the data correlation capability of the HM. This factor addresses the trend toward distributed processing.
3. Hardware monitor compatibility. The target system is required to interface with the logic analyzer previously acquired.
4. Vendor support. Documentation and maintenance of the hardware and software should be available for the target system.
5. Application expenses. The cost of applying the HM to a system is an important consideration. Factors comprising this category include: software monitor development, hardware monitor setup and probe adaptation, and existence of device drivers (or simplicity of design).
6. System type. The target system should be representative of current multiprocessor system architectures, as typified by PLRS (tactical system) or TIS (test system).

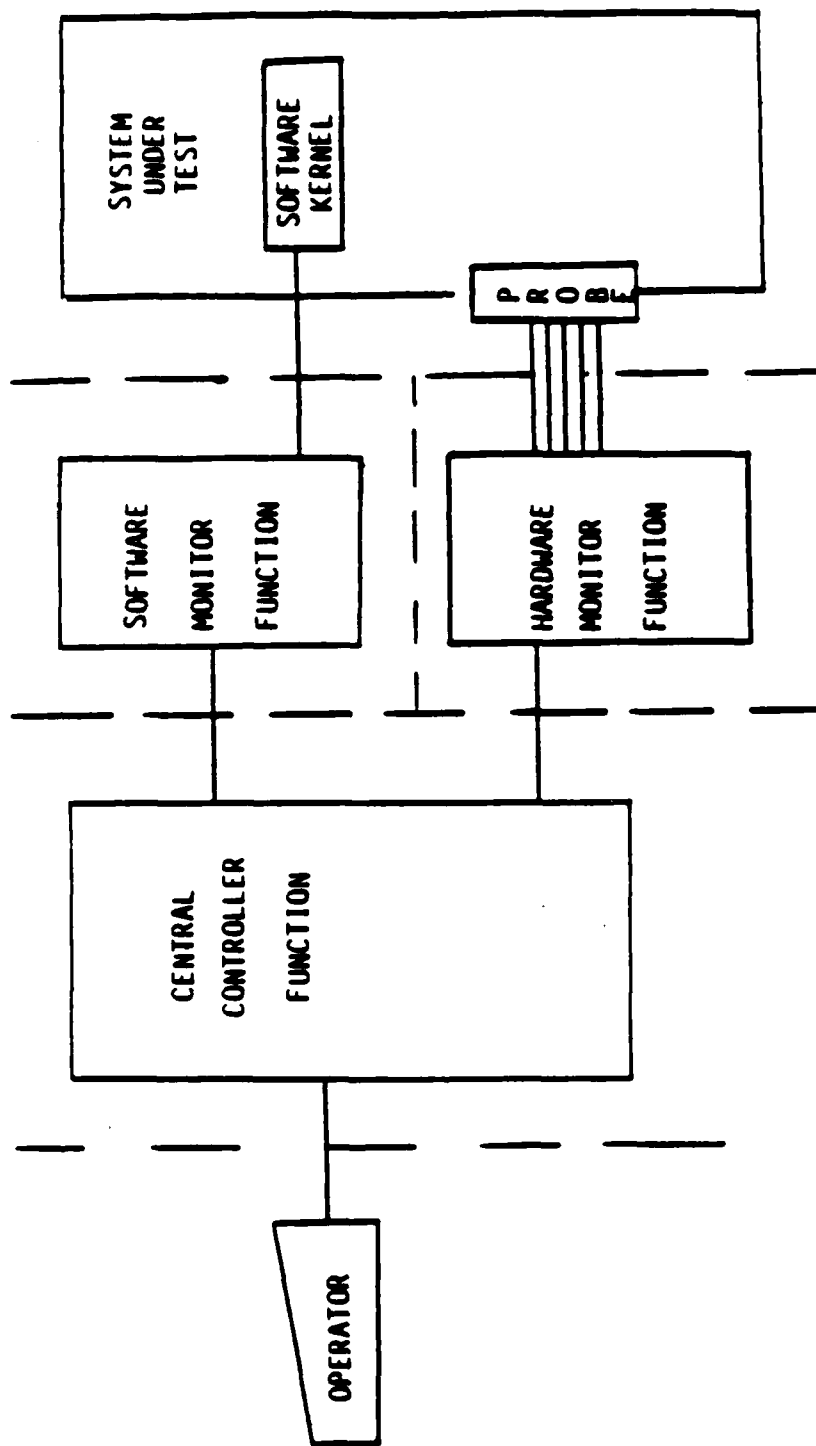


Figure 1. HM Functional Block Diagram

signals are monitored by attaching probes to SUT computer backplanes, by inserting off-the-shelf bus interface cards, or by using emulator capabilities.

These three components work in unison to provide a flexible tool which can be used to extract a wide variety of data from a SUT. In particular, these components allow simultaneous collection of different data from single or multiple processors.

2.2.2 Prototype Hybrid Monitor Configuration

The prototype HM was designed to allow up to ten hardware and software monitors of any mix. (The configuration used for collecting MOPs is depicted in figure 2). The maximum number of monitors used concurrently was two software and one hardware. Two of the items at the bottom of this diagram, TIS post-test analysis and IRIG time code generator, were specific to certain test configurations. The flexibility of the HM architecture was exploited advantageously to convert MOP data (measuring TIS time-stamp accuracy) to a TIS compatible format, allowing the use of existing TIS post-test data reduction and analysis programs. The IRIG (used on TIS) was itself monitored to provide an actual time (within 1ms) of events monitored on the T11 processor. Unfortunately, the DAS does not contain an internal time-of-day clock. When an external clock is required, up to 44 (out of 96) probes are dedicated to provide an event time stamp.

The hardware monitor was interfaced to the HM host (VAX) via an IEEE-488 link and IEU11 device interface. Software monitors and kernels, which are specific to particular MOPs, communicated with the central controller via the VAX/VMS operating system. This was possible since both the HM central controller and software target system were resident on the same hardware, using the virtual features of the VAX/VMS system. Although having the HM central controller and target software system reside on the same computer could have resulted in performance degradation, in operation, the HM was found to use less than ten percent of the system resources, with minimal impact on the monitored software.

2.3 MOP Collection

The HM configuration described above was used to demonstrate the feasibility of collecting twelve MOPs, arranged in eight test configurations. Table II shows the arbitrarily assigned test and MOP numbers and the type of monitor capability demonstrated. Note that hardware and software monitors are on separate processors, demonstrating a multiprocessor capability when executed concurrently.

Application of the HM to the T11 system and VAX software involved defining the HM test configuration (software/hardware monitors, operator displays for real-time inspection of collected data, hardware monitor setups and probe points), initializing and running the data collection, and analyzing the results. The specific MOPs and results of each test case are summarized in appendix G.

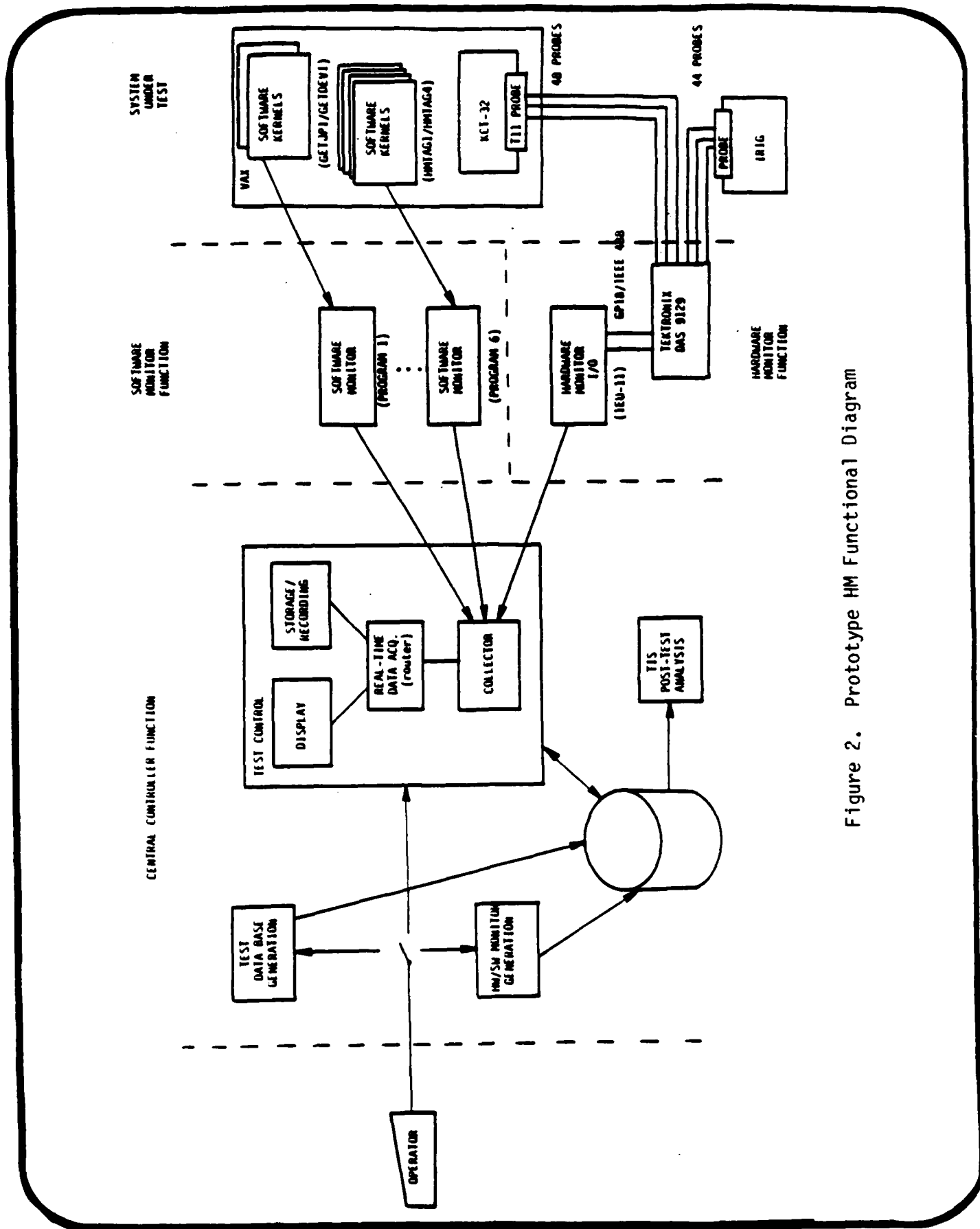


Figure 2. Prototype HM Functional Diagram

Table II. MOP Test Configurations

<u>TEST NO.</u>	<u>MOP(s)</u>	<u>MOP No (s)</u>	<u>MONITOR TYPE</u>
1	Overhead	1	Hardware (HW)
2	Throughput Virtual Memory Overhead Component Response (Delta)	3, 4, 6	2 Software (SW), 1 HW - Hybrid
3	Function Overlap	2	HW
4	Function Trace	8	SW
5	Function Periodicity	10	HW
6	Function Duration Function Processing Time	11, 12	SW (1 monitor with 2 MOPs)
7	Communication Delay (SW) Communication Delay (HW)	16, 17	SW, HW - Hybrid
8	Component Response (Time-of-Day)	18	HW

Table III shows the specific types of MOPs used for prototype validation, grouped by the five categories established for the proposed generic MOPs: system control, memory utilization, component usage, software functions, and network communications. Specific MOPs were chosen for testing based primarily upon the characteristics of the target systems. For example, memory fragmentation was not possible on the target software because of the machine architecture and techniques employed, therefore, no attempt was made to verify this MOP. Instead, virtual memory overhead was used to demonstrate the memory utilization category. Limitations of the prototype HM (only one DAS hardware monitor was available) prevented some measurements, while other limitations were self imposed. No physical modifications or solder connections were made so as not to violate the provisions of the warranty and maintenance contract for the SUT. In actual use for testing, MOPs would also be selected based on target system characteristics, taking into consideration the limitations of the HM.

MOPs were successfully collected for each category of the proposed generic MOPs. Furthermore, the prototype HM was exercised in software, hardware, and hybrid configurations, with the latter demonstrating a multiprocessor capability. Analysis of the test results indicates three areas which affected performance and should be improved for a full-scale HM. They are:

- a. Limitations on the size of the DAS buffer which imposes an upper limit on the number of samples before data must be transferred to the central controller.
- b. Delays in transferring data to the VAX, typically 3 secs and the sequential nature of the acquisition transfer functions. New data cannot be acquired concurrently with transfer to a host.
- c. Lack of an internal clock which required that 44 out of 96 probes be dedicated to monitoring an external clock.

All of the significant limitations of the prototype HM were isolated in the hardware monitor component. Thus, while the HM appears to be a viable concept overall, consideration should be given to examining other hardware monitors for use in a full-scale HM. An examination of recent and future vendor offerings indicates a trend to improve performance in exactly the areas shown deficient. These include larger buffer storage, on board mass storage with concurrent acquisition and data transfer, and integral time-stamp capability.

A limitation of the prototype hybrid monitor exists in the capability to demonstrate multiprocessor hardware monitoring. Another DAS would be needed to properly provide this ability. The presently available equipment comprises a minimal configuration hybrid monitor for demonstrating hardware, software, and hybrid configurations. Concurrent multiprocessor monitoring is achievable to the extent that software and hybrid monitoring can be performed on separate processors.

Table III. Prototype MOP Categories

<u>MEASURES OF PERFORMANCE</u>	<u>PROTOTYPE MOPS No.</u>	<u>MEASURES OF PERFORMANCE</u>	<u>PROTOTYPE MOPS No.</u>
<u>System Control</u>		<u>Software Functions</u>	
Overhead	1	Function Trace	8
Queue Usage		Function Execution Ratio	
Throughput	3	Function Periodicity	10
		Function Duration	11
<u>Memory Utilization</u>		Function Processing Time	12
Memory Fragmentation		Function Overlap	2
Virtual Memory Overhead	4		
<u>Component Usage</u>		<u>Network Communication</u>	
Component Overlap		Host Communication Matrix	
Component Response	6, 18*	Packet Type	
Component Utilization		Packet Size	
		Communications Throughput	3**
		Packet Interarrival Time	
		Channel Acquisition	
		Communication Delay	16, 17***
		Collision Count	

Notes:

* Component response was further decomposed into MOPs for a delta time between events and a time-of-day of an event.

** Data elements used for this MOP were collected by the same monitor used to demonstrate system throughput.

*** These MOPs demonstrated the collection of the same performance measure through different monitor techniques, software (MOP 16) and hardware (MOP 17).

(BLANK PAGE)

APPENDIX A
METHODOLOGY INVESTIGATION PROPOSAL

(BLANK PAGE)

METHODOLOGY INVESTIGATION PROPOSAL

1. TITLE. Computer Hardware/Software Monitor
2. CATEGORY. VISTA, DC³I/Software, Interoperability
3. INSTALLATION/FIELD OPERATING ACTIVITY. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613-7110.
4. PRINCIPAL INVESTIGATOR. Mr. Richard G. Jacques, Software and Automation Branch, STEEP-MT-DA, AUTOVON 879-1957/1870
5. STATEMENT OF THE PROBLEM. Army testers need powerful, flexible and automated methods to examine critical computer hardware and software interaction, access functional expandability of computer hardware/software systems, and to empirically derive simulation model parameters. USAEPG does not currently possess such a capability for both hardware and software monitoring.
6. BACKGROUND. During the past few years, USAEPG has been investigating methods of testing systems containing multiple, interactive processors (7-COR-RD3-EP1-002 and 7-CO-R85-EP0-006). A general purpose hardware/software hybrid monitor was proposed. The concept was not validated during the investigations.
7. GOAL. To validate the proposed method of using a hardware/software hybrid monitor as a tool for testing systems containing multiple, interactive processors, and to verify that hardware/software hybrid monitors can collect the measures of performance (MOP) identified during the previous investigations.
8. DESCRIPTION OF INVESTIGATION.
 - a. USAEPG will connect a brassboard hardware/software hybrid monitor to a target system under test and verify its ability to collect performance information from the target system.
 - b. The US Army Electronic Proving Ground will:
 - (1) Select a target system to test. Candidates include the EDM version of the Position Location and Reporting System (PLRS) and the USAEPG Test Item Stimulator (TIS).
 - (2) Develop necessary software monitors and interface them to the system under test (SUT) and the hardware/software hybrid monitor.
 - (3) Determine probe points for the hardware monitor to connect to the SUT.
 - (4) Collect performance data from the SUT (MOP).

(5) Publish the results, conclusions, and recommendations in a final methodology report.

(6) Publish specifications for a hardware/software hybrid monitor.

9. JUSTIFICATION.

a. Association with Mission. USAEPG responsibilities include developmental testing of complex C-E systems. This system testing necessarily includes the ever increasing number of embedded computers. Hence, it is imperative that current state-of-the-art T&E technology be maintained. This is a relatively new area of testing necessitated by the expanding complexities of testing processor-driven tactical systems. The hardware/software monitors can be used to directly measure the relative efficiencies of hardware/software in accomplishing its tasks. This capability to determine system software performance will prove useful in interoperability testing and thus help address the Army Science Board recommendation to strengthen interoperability testing.

b. Present capability, limitations, improvements, and impact on testing if not approved. USAEPG has purchased a Tektronix DAS 9100 hardware monitor based upon the recommendations of the previous investigations. However, it has not integrated this hardware monitor into any type of hardware/software hybrid monitor. Failure to develop a hardware/software hybrid monitor will preclude adequate testing of complex interoperable systems. Field and laboratory testing of C³I systems will be limited in their ability to evaluate growth capacity of newly developing systems.

c. Dollar Savings. No dollar savings can be assessed at this time. The investigation is being conducted to validate a methodology for a new test capability.

d. Workload. Over the past 5 years, the USAEPG has experienced seven systems which required this type of methodology. Examples of items anticipated for testing are:

<u>System</u>	<u>Test Schedule (FY)</u>				
	86	87	88	89	90
JTIDS	x	x	x		
MCS			x	x	x
RPV	x		x	x	
PLRS	x	x			
IFTE		x	x	x	
SHORAD C2		x	x	x	
JINTACCS		x	x	x	x
PJH	x	x	x		
GPS	x	x	x		
ASAS	x	x	x		
FIREFINDER		x	x	x	x

e. Association with requirements documents. The Army Battlefield Interface Concept (ABIC) outlined the requirements for software/ interoperability testing. The Army Science Board recommends the development of new test tools/methodologies in parallel with developing systems.

f. Others. None.

10. RESOURCES:

a. Financial

	Dollars (Thousands)	
	In-House	FY 86 Out-of-House
Personnel Compensation	36.0	
Travel	4.0	
Contractual Support		181.0
Material & Supplies	1.0	
Equipment (Lease)		
General and Administrative Costs	4.0	
SUBTOTALS	45.0	181.0
FY TOTALS	226.0	

b. Explanation of Cost Categories.

(1) Personnel Compensation. This represents compensation chargeable to the investigation for civilian labor.

(2) Travel. Approximately four trips will be used to visit contractor's facility or other Government installations.

(3) Contractual Support. Most of the work will be accomplished by contractor personnel.

(4) Materials and Supplies. This investigation will involve normal administrative costs.

(5) G&A Costs. Included in contractual support.

c. Obligation Plan.

	FY86				
Obligation Rate (Thousands)	1	2	3	4	Total
	192	12	11	11	226

d. In-House Personnel.

(1) In-House Personnel Requirements by specialty.

	NUMBER	MAN-HOURS FY86		TOTAL
		REQUIRED	AVAILABLE	REQUIRED
EE, GS-855	1	900	900	900
CS, GS-334	1	900	900	900

(2) Resolution of nonavailable personnel. Not applicable.

11. INVESTIGATION SCHEDULE.

	1986											
	O	N	D	J	F	M	A	M	J	J	A	S
In-House	-	-	-	-	-	-	-	-	-	-	-	R
Contract	A	:	:	:	:	:	:	:	:	:	:	:

Symbols:

- - - Active Investigation Work (All Categories)
 . . . Contract Monitoring (In-House Only)

A Award of Contract
 R Final report at Headquarters, TECOM

12. ASSOCIATION WITH TOP PROGRAM. Present TOPS regarding software may require changes relative to data acquisition, reduction and analysis.

FOR THE COMMANDER:

(signed)
 JOHN R. SUTHERLAND, JR.
 LTC, AD
 Director, Materiel Test Directorate

APPENDIX B
REFERENCES

(BLANK PAGE)

REFERENCES

1. Methodology Investigation Final Report Software Testing of Multiprocessor Systems, dated October 1985. TECOM Project No. 7-CO-R85-EPO-006. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613.
2. Methodology Investigation Final Report Software Testing of Multiprocessor Systems, dated October 1984. TECOM Project No. 7-CO-RD0-EP1-002. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613.
3. General System Specification for the Hybrid Monitor, 15 January 1985. Document No. SS-01-01. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613.
4. Development Specification for the Hybrid Monitor, 15 January 1985. Document No. DS-01-01. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613.
5. Technical User's Manual for Hybrid Monitor, September 1986. Document No. TUM-01-01. U.S. Army Electronic Proving Ground, Fort Huachuca, Arizona 85613.

(BLANK PAGE)

APPENDIX C
ACRONYMS AND ABBREVIATIONS

ABIC.....Army Battlefield Interface Concept
 ASAS.....All Source Analysis System
 C³I.....Command, Control, Communications, and Intelligence
 C-E.....Communications Electronics
 CPU.....Central Processing Unit
 DAS.....(Tektronix) Data Acquisition System
 DC³I.....Distributed C³I
 DRA.....Data Reduction and Analysis
 DT.....Developmental Test
 EDM.....Engineering Development Model
 FY.....Fiscal Year
 G&A.....General and Administrative
 GPIB.....General Purpose Interface Bus
 GPS.....(see NAVSTAR GPS)
 HM.....Hybrid Monitor
 HW.....Hardware
 ICE.....In-Circuit Emulator
 IFTE.....Intermediate Field Test Equipment
 I/O.....Input/Output
 IRIG.....Inter-Range Instrumentation Group
 JINTACCS.....Joint Interoperability of Tactical Command
 and Control Systems
 JTIDS.....Joint Tactical Information Distribution System
 LAN.....Local Area Network
 MCS.....Maneuver Control System
 MOP.....Measure of Performance
 NAVSTAR GPS...Global Positioning System
 NBS.....National Bureau of Standards
 PJH.....PLRS/JTIDS Hybrid
 PLRS.....Position Location Reporting System
 PROM.....Programmable Read-Only Memory
 ROM.....Read-Only Memory
 RPV.....Remotely Piloted Vehicle
 SHORAD C².....Short-Range Air Defense, Command and Control
 SMI.....Soldier Machine Interface
 STMS.....Software Testing of Multiprocessor Systems
 SUT.....System Under Test

SW.....Software
TECOM.....U.S. Army Test and Evaluation Command
T&E.....Test and Evaluation
TIS.....Test Item Stimulator
TOP.....Test Operations Procedure
USAEPG.....U.S. Army Electronic Proving Ground
VAX.....Virtual Address Extension
VISTA.....Very Intelligent Surveillance and Target Acquisition System
VMS.....Virtual Memory System
V&V.....Verification and Validation

APPENDIX D

PROTOTYPE HARDWARE MONITOR CHARACTERISTICS

(BLANK PAGE)

1.0 TEKTRONIX DAS 9100

a. The Tektronix 9100 series digital analysis system consists of the color 9129 mainframe, optional data acquisition modules, and pattern generator modules. It can be remotely controlled via an RS-232 or General Purpose Interface Bus (GPIB) interface. Exact characteristics are dependent upon the options and configuration in use. Manuals should be referenced for the capability of a particular configuration.

b. The data acquisition modules enable up to 96 channels of information to be monitored and up to 4096 bits of data stored in a buffer memory. The data acquisition module (91A24 series), which permits disassembly of micro-processor code, has a maximum sampling rate of 10 MHz and a buffer memory depth of 1024 bits. Trigger qualifiers are programmable, and accept up to three external inputs.

c. Many personality modules are available for the disassembly of micro-processor and minicomputer code. Probes are provided to monitor address and data busses. The 9100 does not provide emulation, however, this function is provided by other Tektronix products.

d. Precision event tracing and capture is provided through five independent word recognizers using two operational modes. The word recognizers provide up to 16 levels of sequential event tracing plus data qualification for selective data storage.

(1) In one mode of operation, data qualifier words are available for starting and stopping data storage. The 16-level sequential stack is available for trigger tracing of qualified data. Also, the occurrence counter, trace triggers counter, and 100 ns timer are available with the stack; a SYNC output signal can be enabled at each level. Also, a parallel "OR" trigger RESET event recognizer is available for resetting the stack level to level one.

(2) The second mode provides four word recognizers for logical data qualifier combinations, two word recognizers for enabling data storage and two word recognizers for disabling data storage. The 16-level sequential stack is also available for trigger tracing of qualified data.

e. Two event counters and a 100 ns resolution timer combine with triggering to help track software in complex digital systems. Events can be counted up to 4096 occurrences per level and then reset if necessary. At each sequential trigger level a SYNC pulse out can be programmed to trigger or strobe external equipment.

f. An External Trigger Enable input allows the 91A24 modules to be automatically "armed" from an external source to look for a trigger on incoming data only when requested.

g. The 91A24 modules can be used to track down intermittent faults with the auto comparison between reference memory and the 91A24's data acquisition memory. And data differences can be displayed in color with user-defined mnemonic disassembly to help identify anomalies. Also provided is the ability to perform column masking in conjunction with a programmable compare window.

h. Up to 80 channels of programmable output are available with the 25 MHz pattern generator module. With output clocks, up to 10 separately programmable strobes are available. Subroutine nesting up to 16 levels, external interrupts, single-key functions, and selectable Radix are featured in the pattern generator module.

i. The 9100 is equipped with a 160K byte cartridge tape drive. Remote and master/slave operation is provided by an RS-232 interface. All instrument status, including both menu information and acquisition and pattern generator memory contents can be obtained via this data link.

APPENDIX E
MEASURES OF PERFORMANCE

(BLANK PAGE)

1.0 MEASURES OF PERFORMANCE

1.1 Introduction

Many aspects of a system's performance may be analyzed. This section is concerned with measures associated with verification of system performance. The MOPs described herein have been divided into five basic categories, as delineated in table IV.

1.2 System Control

The first category, system control, consists of those aspects related to the overall control of the system. Most systems employ some mix of tables, queues, timers, and counters to control their processing both at the executive and application levels. It is important for the tester to be able to observe the activity and correlate these parameters to determine if system performance is within specifications.

1.2.1 Overhead

This is a measure of the percentage of CPU processing time required by the executive program to control a processor's applications programs. Examples of executive processing which constitute overhead are task scheduling, memory allocation, and message queuing. Measuring overhead provides one means for the tester to determine if adequate CPU resources are available for the application.

1.2.2 Queue Usage

a. This measurement provides a means of evaluating the management of system queues. It also provides a method of evaluating the effect of loading on the various queues and their effect on system performance.

b. Queue usage is a necessary measure for the tester, as it can provide an immediate indication that data is being lost. The tester can also use this measure to determine if queued data is being processed at the required rate. Additionally, it can be determined if queues have been improperly prioritized or have excessive/insufficient allocated memory (e.g., a queue allocated 10 locations is never observed to use more than one). This measure is normally analyzed in an effort by the testing organization to confirm compliance with specifications.

1.2.3 Throughput

a. This measurement identifies the amount of processing completed per unit of time at a system level.

b. A measurement of this nature is required to determine the overall impact of different loads on the system's performance. Throughput provides an overview from which conclusions can be drawn as to which functions are adversely affecting the system. This measure provides the higher level overview necessary to direct testing efforts if a problem is encountered.

Table IV. Measures of Performance

Measures of Performance	VV	IT	HFL	SD	DT
SYSTEM CONTROL					
Overhead	X	X			X
Queue Usage	X	X		X	X
Throughput	X	X			X
MEMORY UTILIZATION					
Memory Fragmentation	X	X		X	X
Virtual Memory Overhead	X	X	X		X
COMPONENT USAGE					
Component Overlap	X	X	X		X
Component Response (including interrupts)	X	X	X		X
Component Utilization	X	X	X		X
SOFTWARE FUNCTIONS					
Function Trace	X	X	X	X	X
Function Execution Ratios	X	X		X	X
Function Periodicity	X	X		X	X
Function Duration	X	X		X	X
Function Processing Time	X	X		X	X
Function Overlap	X	X		X	X

VV = Verification/Validation

IT = Integration Testing

HFL = Hardware Fault Location

SD = Software Debug

DT = Developmental Testing

Table IV. Measures of Performance (continued)

Measures of Performance	VV	IT	HFL	SD	DT
NETWORK COMMUNICATION					
Host Communication Matrix	X	X		X	X
Packet Type	X	X		X	X
Packet Size	X	X		X	X
Communications Throughput	X	X		X	X
Packet Interarrival Time	X	X		X	X
Channel Acquisition Delay	X	X		X	X
Communication Delay	X	X		X	X
Collision Count	X	X		X	X

VV = Verification/Validation

IT = Integration Testing

HFL = Hardware Fault Location

SD = Software Debug

DT = Developmental Testing

1.3 Memory Utilization

a. Many memory storage management schemes are in existence today. Most of these schemes fall into one of the seven following types:

- . Single contiguous memory.
- . Partitioned contiguous memory.
- . Relocated partitioned contiguous memory.
- . Paged memory.
- . Demand-paged memory.
- . Segmented memory.
- . Segmented and demand-paged memory.

b. The MOPs in this category are designed to measure those system parameters associated with memory management.

1.3.1 Memory Fragmentation

Memory fragmentation is the result of allocating and deallocating dynamic storage space in a manner such that larger contiguous blocks of memory are not available when required. This means that at a given time there may be enough total free dynamic memory to support a memory request but only as several small non-contiguous blocks. If this occurs, a requester must wait until a large enough contiguous block becomes available. This condition can seriously impact system performance and therefore warrants monitoring by the tester.

1.3.2 Virtual Memory Overhead

Program execution time can be adversely affected by the utilization of virtual memory management. The time spent in such activities as swapping working sets to secondary storage add to the system overhead and actual program execution time. The time allocated to management of virtual memory can be measured to see if it is excessive or whether certain software functions should be permanently resident.

1.4 Component Usage

A component or resource, used in this context, is a physical entity within a computer system. It may include a CPU, I/O controller, tape drive, disk, etc. In particular, in a multiprocessor network environment, it most certainly includes other processors. That is, any one processor may be treated as a component of the system.

1.4.1 Component Overlap

a. This is a measure of the percent of time system components operate in parallel.

b. This measure is useful in a test environment because it allows the detection of blockage between two or more components of a system. Operation of a system component is sometimes specified in terms of non-interference with other components. Adherence to specifications can be confirmed using this measure. The tester can further use this measure to determine probable system bottlenecks by identifying non-overlapping components and system load scenarios which induce non-overlap conditions.

1.4.2 Component Response

a. This measure involves determining the amount of time required to complete processing of requests by system components.

b. This measure is important to the tester to insure that components respond within the specified time. The tester must also be concerned with the effects of multiple processes and peripheral contention in a multiprocessor environment. Varying load testing may be performed to accurately determine saturation points of the various components of a system. This type of testing is also necessary to evaluate the effects of response times that are within specifications but are slower than the unloaded nominal response time.

c. While component response would normally be used by the tester in the context of peripheral or intercomputer responses, it is also used to address intracomputer component response times. Intracomputer response times include the time required by a CPU to respond to an interrupt issued by another component. It is important for the tester to be able to address these intracomputer response times to determine if the method used to process these interrupts is inducing excessive delays.

1.4.3 Component Utilization

a. This measure is employed to evaluate the usage and resource requirements of components in a system.

b. An accurate measure of CPU time must be obtained to determine if the specified reserve CPU capacity has been met. In a multiprocessor environment, this measure can be used to determine an imbalance in the workload. This would allow the tester to suggest shifting this load to bring a system's performance within specification.

c. The tester must also determine if a peripheral is performing within specification or if the utilization of a peripheral is impacting system performance beyond acceptable limits. This measure can additionally be used to suggest corrective action if the underuse of a peripheral indicates that part of its processing time or buffer memory can be allocated to other processes.

d. Component utilization can also be used to determine if adequate data transfer reserves are being maintained. This applies to the update rate of files, interprocessor communications, and to data transfer loads on I/O controllers.

1.5 Software Functions

This section addresses those measures related to software functions. A function can be defined as a single instruction, a short sequence of instructions, a module (procedure), a set of modules (task), or any combination of these which define some unique system processing. Within a multiprocessor system, this sequence can span processors for those functions which require interprocessor interactions.

1.5.1 Function Trace

a. This measure allows the determination of correctness by showing that an input data item has followed an expected path through the system and has produced the expected output. Additionally, results are used to indicate which software elements have been executed.

b. The tester is concerned with the ability to prove the correctness (or incorrect performance) of a SUT. A major component of correctness in computer systems is the production of an expected output from the application of a known input. A method often employed to prove correctness is the tracking of an input through the system, inspecting any pertinent intermediate results, and confirming that the expected output is produced. This is sometimes referred to as data flow analysis.

c. Another use of function tracing is to provide a history of the control flow. This information allows the tester to confirm that proper segments of a program are executed in the correct order. Depending upon the level of tracing, various levels of test coverage (or thoroughness) may be derived from function tracing. This tracing gives the tester a quantitative measure of the completeness of testing.

1.5.2 Function Execution Ratios

a. This measure uses the occurrence of a function correlated with time or inputs to determine if a function is executing at the required rate.

b. This measure is essential to the tester to confirm that all processing required is being performed in a timely fashion. This measure can also be used in conjunction with saturation testing to confirm that nonessential functions are being suspended to allow critical functions to be completed (i.e., priorities are correctly assigned).

c. Since functions can be single instructions, this MOP could be used for an instruction mix analysis. Such an analysis is frequently helpful in uncovering excessive use of time-consuming instructions. By examining operation codes, frequency of instructions in a particular segment can be tallied for comparisons.

1.5.3 Function Periodicity

This measure, which provides information pertaining to the differential time between executions of a function, is important to the tester to determine if time-critical periodic processing is being performed at the required intervals. It is important that a monitor be able to extract, correlate, and

present this information so that the tester can evaluate the influence of loading and other factors on this measure.

1.5.4 Function Duration

a. This measure is the total time required to complete a software function.

b. It is important the tester evaluate the response of the system as it is subjected to various loads. As an example, a tracking function could perform as specified in a light load scenario, but may be producing erroneous outputs under a heavier load. If this tracking function was defined by several modules, errors may not be detectable at the module level, as each module could complete processing within its specified time. The interference induced by increased executive processing (overhead) required by heavier loads could result in scheduling delays of modules which comprised the entire tracking function. In this case, the problem could be quickly identified by function duration time measurements.

1.5.5 Function Processing Time

a. This is a measure of the CPU time required by a function to complete its processing. This measure excludes the executive processing time (overhead) required to control the processing of a function.

b. Function processing time supplements the information available from an analysis of function duration. This measure can be used to determine the degree to which total function time consists of individual module CPU execution times.

1.5.6 Function Overlap

a. This measure correlates concurrent software functions.

b. This measure can be used to determine if functions are blocking one another, resulting in possible deadlock conditions, or if functions which should execute in sequence are properly synchronized.

1.6 Network Communication

The network communication MOPs were derived from an examination of the National Bureau of Standards (NBS) monitor. These MOPs are applicable to multiprocessor configurations where LANs provide the communication paths among processors. Because of the overlap between communication link monitors and digital message test drivers, such as the TIS, these MOPs were not fully analyzed but are presented below for completeness.

1.6.1 Host Communication Matrix

This measure indicates the traffic flow between connected components of the SUT.

1.6.2 Packet Type

This measure indicates the distribution of each type of packet transmitted.

1.6.3 Packet Size

This measure records the number and proportion of data packets of particular length classes.

1.6.4 Communications Throughput

The communications throughput measure determines the actual rate of message transfer on the communications channels. Also included is the channel capacity used for overhead and for retransmission of messages received in error. Other related statistics on message error rate, bit error rate, channel availability, etc., would be available for analysis as a result of the data elements required to derive the communications throughput measure.

1.6.5 Packet Interarrival Time

This measure indicates the number of packet interarrival times which fall into particular time classes. An interarrival time is the time between consecutive carrier (network busy) signals.

1.6.6 Channel Acquisition Delay

This measure records the times spent by components contending for and acquiring the channel. A channel acquisition delay begins when a component becomes ready to transmit a packet and ends when its first bit is transmitted onto the channel. Included is all of the time spent deferring due to a busy channel and the time recovering and backing off from one or more collisions.

1.6.7 Communication Delay

This measure indicates the delays that components incur in communicating packets to their destinations. A communication delay begins when an original packet becomes ready for transmission and ends when that packet is received by the destination (which may be several transmissions later). One communication delay exists for each packet communicated. This delay may include several channel acquisition delays.

1.6.8 Collision Count

This measure tabulates the number of collisions a packet of any type encounters before completion of a successful transmission.

APPENDIX F
HYBRID MONITOR METHODOLOGY

(BLANK PAGE)

1.0 INTRODUCTION This appendix describes the development of the HM methodology, which led to the concept of integrating the capabilities of both hardware and software monitors controlled by a central controller--the HM approach--in an attempt to exploit the desirable features of both monitors while minimizing their shortcomings.

1.1 Hybrid Monitor Methodology The HM methodology progressed from the basic test methodology to an analysis of the tools suitable for multiprocessor testing. An analysis of the tool features was used to select the most viable approach to meet the investigation objectives.

1.1.1 Test Methodology

a. A method for approaching the testing of software is a performance analysis. A performance analysis defines those characteristics of the system which can be used to determine correctness of processing. These characteristics include: the responsiveness of the system under various loads, the concurrency of processes within the system, and the capacity of the system.

b. These characteristics can be determined by obtaining MOPs from the SUT. These MOPs are addressed in appendix E. It should be noted that the MOPs intentionally overlap one another. To define MOPs in terms of strict non-overlapping measures is not possible if the measures are to be applied to a variety of systems and implementations.

c. In order to extract the measures, a testing organization must have tools at its disposal which are capable of performing the measurements which the tester deems necessary to prove correctness.

d. The most widely used tools for performing these measurements are hardware monitors and software monitors. A newer approach, the HM, is potentially a flexible tool capable of gathering a wide variety of data. These tools are described below with the methods of data collection used.

e. All of the MOPs address the resources of a system. The resources of a system are memory, processors, devices, and information. Memory refers to the high speed storage of computers. Memory types include read/write, ROM, and PROM. Processor types include CPUs and I/O controllers. Devices include disks, tape drives, and user terminals. Information refers to the non-hardware portion of a system. Information includes messages, files, data, applications programs, and executive programs.

f. The structural outline of a performance analysis is presented in figure 3.

1.1.2 Multiprocessor Software Monitor

a. A software monitor is a test tool which is inserted into the program structure of the SUT. In a multiprocessor environment, this type of monitor is generally implemented as an external processor (one which is not a part of the SUT) which communicates with kernels of code embedded in each of the

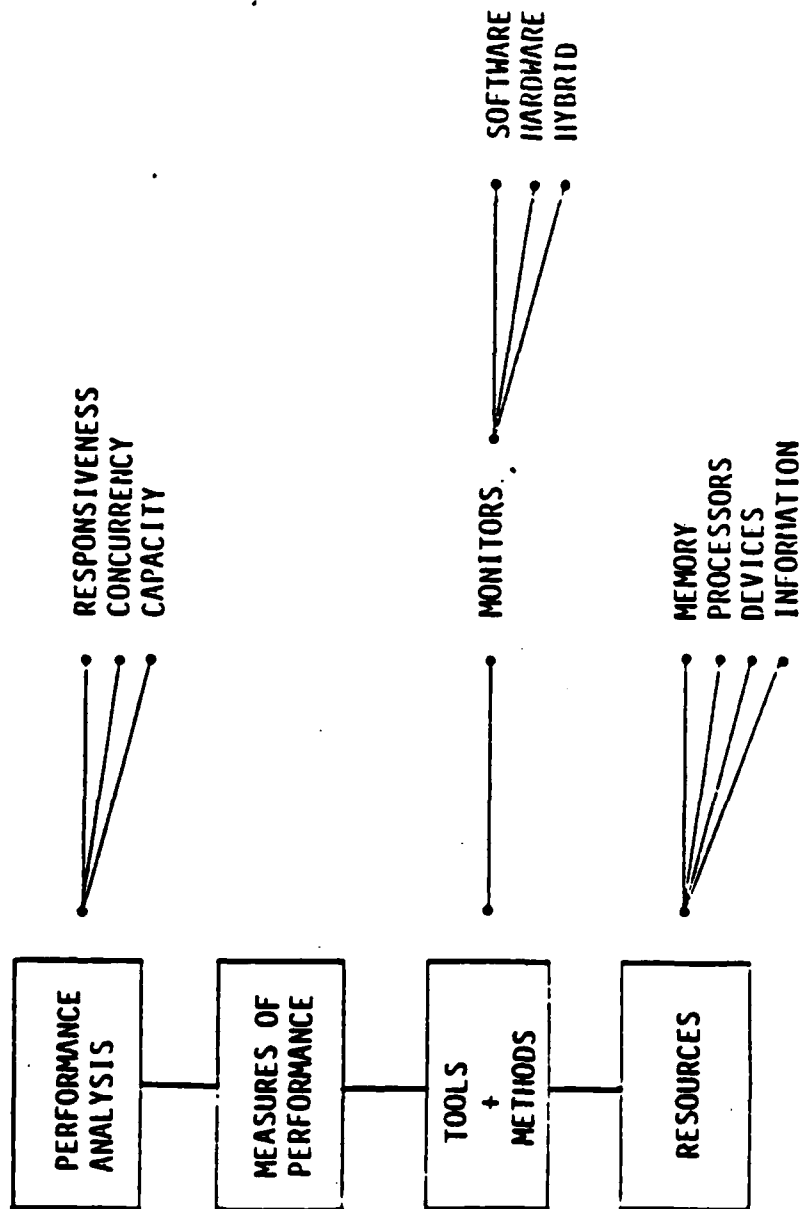


Figure 3. Performance Analysis Methodology

processing units which comprise the test system. These kernels accept commands from the test operator via I/O from the external processor. Commands are interpreted, and the requested data is gathered and sent back to the external computer via an I/O channel.

b. This test tool requires that the SUT be capable of supporting the kernel's resource requirements (memory, CPU time, I/O port, I/O handling). A tool of this type is intrusive, for it must use the resources being observed by the tester. Therefore, the tester must know the impact of the software monitor on the system.

c. An example of a multiprocessor configuration for a software monitor is shown in figure 4.

1.1.3 Multiprocessor Hardware Monitor

a. A hardware monitor is a device which is used to sense changes in the state of the hardware components that comprise a SUT. This sensing is carried out by attaching probes to the hardware at test points which provide signals indicating changes of state. These probes are used to monitor the address bus, data bus, I/O line signals, etc. These signals are correlated using timers, counters, and comparators. The data collected is formatted and presented in various displays or stored for post-processing.

b. A hardware monitor is generally non-intrusive, for it does not draw on any of the resources of the SUT. This type of monitor is limited by the accessibility of the required test points, the number of probes available to collect the data, and the complexity of defining the events to be monitored.

c. An example of a multiprocessor configuration for a hardware monitor is shown in figure 5.

d. A hardware emulator is a device commonly thought of as a hardware monitor but in actuality far exceeds a monitor's capability. The idea behind an emulator is that the CPU is replaced with a device that meets the same performance of the CPU, plus allows viewing the internal workings of the CPU. Such items as registers, flags, and instruction execution can be closely monitored in real time. When halted, this type of device permits changing registers and flags, thereby allowing a variety of reconfigurations of the program state.

e. With current technology, most emulators are associated with microprocessor applications. Under this application, they are commonly called in-circuit emulators (ICE). Most microprocessors are marketed with an associated ICE available for use in the software development and hardware/software integration. Use of hardware emulators outside of microprocessors is limited due to the enormous development cost. For example, a hardware emulator for a VAX would probably rival, if not exceed, the development cost of the VAX itself.

1.1.4 Multiprocessor Hybrid Monitor

a. A HM is the combination of a software and a hardware monitor which are used in unison to collect and correlate data from a SUT.

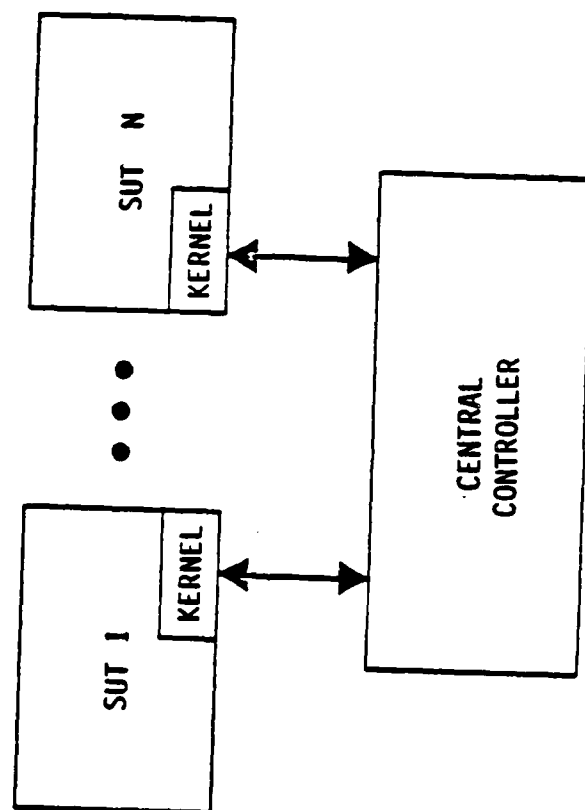


Figure 4. Software Monitor Configuration

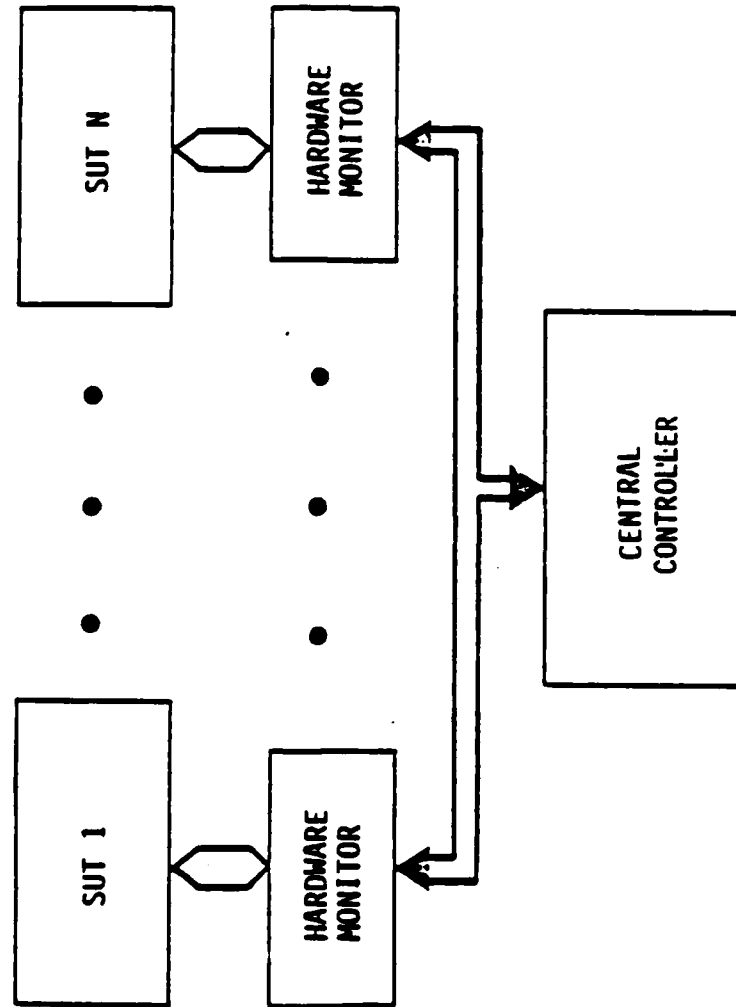


Figure 5. Hardware Monitor Configuration

b. As a hybrid contains both types of monitors, it can be applied to any test configuration which could be addressed by the individual monitors. The ability to blend the qualities of the two monitors also allows the hybrid to address test problems which could not be approached using either of the monitors individually.

c. A hybrid expands the event definition capabilities available to the tester by combining the abilities of both types of monitors. This combination also allows the elimination of some undesirable features of the individual monitors. Detecting certain events with software may remove a requirement to physically open a processor's container to allow access with hardware probes. Using a hardware monitor to detect the flagging of an event processed by a software monitor, greatly reduces the software monitor's impact on the test system as I/O is no longer required to relate occurrences to the external processor.

d. A test situation which required the complex logic capabilities of a software monitor and the high sampling rates attainable with a hardware monitor could be addressed using a hybrid as follows:

- (1) The software monitor's kernel program is used to perform the complex logic.
- (2) The result of the logic performed is stored into a memory word of the SUT.
- (3) The hardware monitor is used to detect the storing of this memory word.
- (4) The hardware monitor extracts this word when stored and processes it or passes it to the software monitor's external computer for processing.
- (5) The hardware monitor is used, at the same time, to collect timing information on the software monitor's processing. This allows the latter's impact on the test system to be accurately assessed.

e. This blending of capabilities can span the full spectrum from exclusive use of the software monitor to exclusive use of the hardware monitor to allow the tester to extract the required data from the SUT.

f. An example of a multiprocessor configuration for a HM is shown in figure 6.

1.1.5 Comparison of Monitor Characteristics

a. A comparison of the three types of monitors was made based on characteristics considered desirable for a general purpose tool. The monitors were evaluated on their ability to display the characteristics over a wide variety of applications. The three monitor types (hybrid, software, and hardware) are shown in table V, with the results of the evaluation dichotomized into strong and weak.

b. Examination of the characteristics of the monitors shows clearly, in the hybrid approach, the synergy of an integrated software and hardware

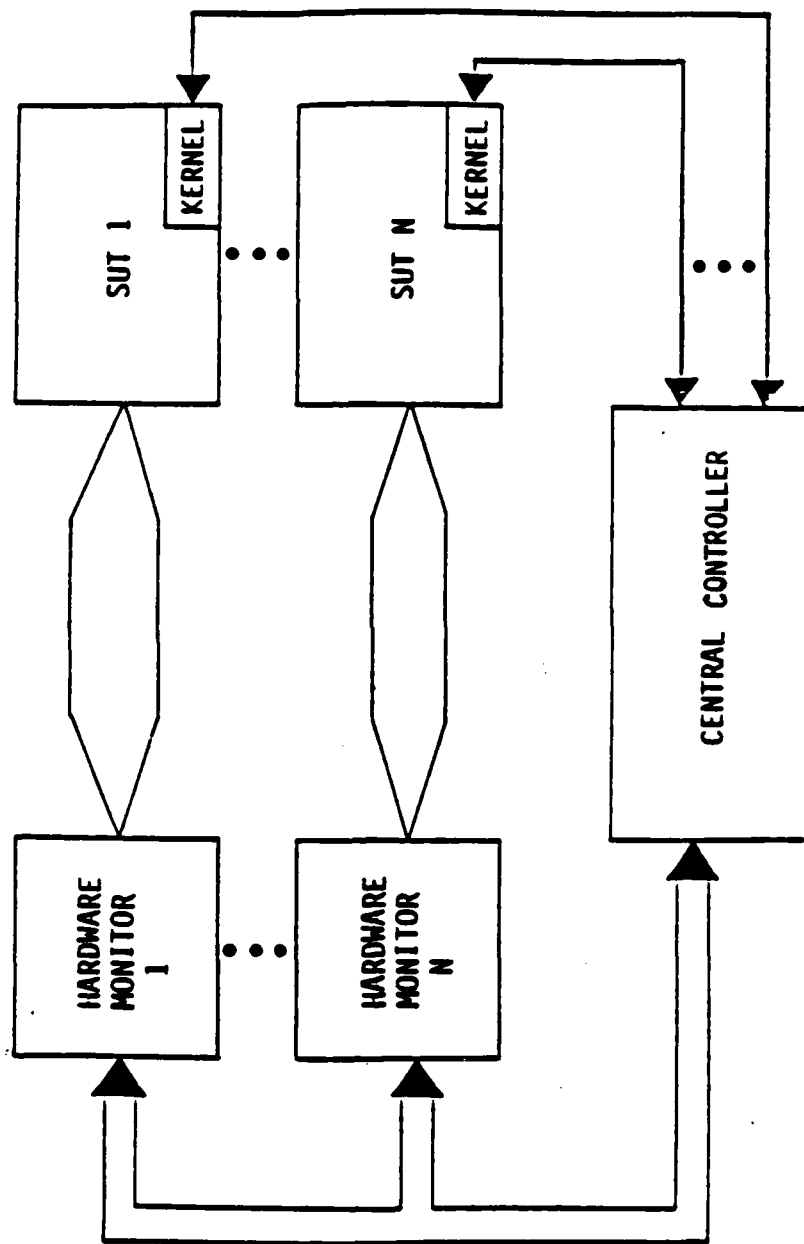


Figure 6. Hybrid Monitor Configuration

Table V. Comparison of Monitor Characteristics

<u>HYBRID</u>	<u>SOFTWARE</u>	<u>HARDWARE</u>	<u>CHARACTERISTIC</u>
S	S	W	Highly complex logic functions
S	W	S	Does not require test system resources
S	S	W	Able to control/be controlled by test systems
S	S	W	Access to any processor memory
S	W	S	Access to data which is not available through processor's instruction set
S	S	W	Able to monitor software not stationary in memory
S	W	S	High timer resolution/event rate
S	S	W	Able to correlate maximum number of inputs
W	W	S	Portability
S	W	S	Able to monitor events on external devices
W	W	W	Is useful without intimate knowledge of test system

S: Strong
W: Weak

architecture. The only drawbacks to a HM, for the features examined, are the lack of portability and the expert knowledge required. Because of the advantages that the hybrid approach offers over stand-alone software or hardware monitors, the HM concept was chosen for further design and development efforts.

APPENDIX G
SELECTED MOP RESULTS

Test No. 1.

Monitor type(s) demonstrated: HW.

Description: Used to demonstrate the collection of MOP1 on a single processor target system.

Test Configuration.

Software monitor: None

Hardware monitor (MOP 1): DAS 9100 with probes attached to the T11 processor of the KCT.

Test No. 1 MOP No. 1

MOP Category.

Primary: System Control

Secondary: Overhead

Specific MOP.

Name: T11 software overhead

Description: Measurement of the amount of overhead that the T11 software takes to handle a transfer request.

Data elements collected: A delta time element was collected from the DAS. The delta time is a measurement of the time the T11 software takes from the acceptance of the transfer packet to the queuing of the transmit packet to the I/O processor.

Results.

The delta time was collected and displayed on the users terminal in real time at four second intervals. Resolution of the delta time was 100 ns. The delta times were also logged to a disk file.

The overhead was found to vary with buffer lengths and traffic loads. Overhead did not vary with data transfer rates.

Limitations/Special Considerations.

The hardware monitor had limited capability for a real time analysis of this MOP. For each delta time returned, the DAS-VAX overhead was approximately 4 seconds. The majority of this time (about 3 secs.) was spent waiting on the DAS service request. This test is adequate for statistical sampling or if the monitored event periodicity exceeds the 3 second delay.

Test No. 2.

Monitor Type(s) Demonstrated: 2 SW, 1 HW - Hybrid

Description: This test demonstrated a hybrid configuration on separate target processors. Also, a capability of controlling multiple monitors (3 total) with the central controller was shown. This test collected MOPs 3, 4, and 6.

Test Configuration.

Software monitor (MOP3): Two software kernels were used for this MOP. Both kernels were embedded within the application program. The kernels are designed to collect the transmit count and the CPU busy time. The monitor collects the data from the kernels as the data becomes available and then passes the information to the central controller. The central controller displays the data on the users terminal and logs the data into a disk file.

Software monitor (MOP4): A single software kernel is used for this MOP. The kernel was embedded within the application program at the point where the MCS application transmits the message packets. The kernel takes advantage of the VAX VMS in using the system service routine GETJPI. The monitor collects the data from the kernel and passes it to the central controller where it is displayed on the users terminal and logged in a disk file.

Hardware monitor (MOP6): DAS 9100 with probes attached to the T11 processor of the KCT.

Test No. 2 MOP No. 3

MOP Category.

Primary: System Control (Network Communication)

Secondary: Throughput (Communications Throughput)

Specific MOP.

Name: MSCPQT throughput.

Description: Measurement of the number of transmit messages that the MCS application can transmit within ten minutes.

Data elements collected: Specific elements collected from the application were transmit count per time period, CPU time busy per time period, accumulated CPU time busy, and accumulated transmit count.

Results.

The kernels successfully kept track of the transmit count and the CPU time busy. The monitor collected this information from the kernels every ten seconds and passed the following information to the central collector: transmit count per time period, CPU busy time per time period, accumulated transmit count, and accumulated CPU busy time. Resolution of the CPU time was ten milliseconds.

The results showed that the number of transmit messages processed per time period were dependent on system usage. With light usage six to seven messages could be processed within the time period. With heavy usage four to five messages could be processed. CPU time was consistent with the amount of messages processed.

Limitations/Special Considerations.

Detailed knowledge is required to implement the kernels to collect the pertinent information. An element such as CPU busy must be implemented in such a way as to collect the amount of time that a process is allocated to a processor. If this information is not available then the implementer must have detailed knowledge to modify the operating system to gather such information. For this test the VAX/VMS system service GETJPI was used to collect the CPU busy time. The transmit count was implemented in the kernel without use of a system service call.

Special note should be made that this MOP demonstrates the collection of throughput for both system control and network communication MOP categories.

Test No. 2 MOP No. 4

MOP Category.

Primary: Memory Utilization.

Secondary: Virtual Memory Overhead (VMO).

Specific MOP.

Name: Virtual Memory Overhead (VMO) of the MCS application.

Description: Measurement of the MCS applications virtual memory overhead to show that the applications transmit rate is not adversely affected by virtual memory overhead.

Data elements collected: The software kernel embedded in the application returns: working set size, page fault count, active page table count, process page count in working set, global page count in working set, paging file quota, current paging file usage, working set size quota, peak virtual size, peak working set size, free page count, and page fault quota.

Results.

The software kernel successfully extracted the pertinent data during initialization of each transmission message. The monitor collected the data every ten seconds then formatted the data and passed the data to the central controller. The central controller displayed the data in real-time and logged the data to a disk file.

The terminal display and the log file showed that VMO was non-existent when the application was actually transmitting the message packets. Virtual memory overhead was high at 484 page faults when the application was in initialization phase.

Limitations/Special Considerations.

The elements required for measuring VMO are most easily obtained when available from the operating systems memory manager. If this information is not available then the implementer must have detailed knowledge and resources available to modify the operating system to gather such information. For this test the VAX/VMS system service GETJPI was used to collect the VMO elements.

Test No. 2 MOP No. 6

MOP Category.

Primary: Component Usage.

Secondary: Component Response (Delta Time).

Specific MOP.

Name: KCT32 Response Time.

Description: Measurement of the time that the KCT32 takes for processing a transmit message.

Data elements collected: A delta time element is collected from the DAS with a resolution of 100 ns. The delta time is a measurement of the time from when the T11 software receives a request to process a transmit message to the T11 completion of the transmission packet.

Results.

The delta time was collected and displayed on the users terminal in real time at four second intervals. The delta times were also logged onto a disk file.

Component response was found to vary with buffer lengths, traffic loads, and with data transfer rates.

Limitations/Special Considerations.

The hardware monitor has limited capability for MOPs of this type. For each delta time returned the DAS-VAX overhead was approximately 4 seconds. The majority of this time (about 3 secs.) was spent waiting on the DAS service request. This test is adequate for statistical sampling or if the monitored events occur at a low rate.

Test No. 3.

Monitor type(s) demonstrated: HW

Description: This test setup was used to collect MOP2.

Test Configuration.

Software monitor: None.

Hardware monitor (MOP2): DAS 9100 with probes attached to T11 processor of KCT. Additional probes were connected to an IRIG time code generator to obtain a time stamp for the monitored events.

Test No. 3 MOP No. 2

MOP Category.

Primary: Software Functions.

Secondary: Function Overlap.

Specific MOP.

Name: Function overlap of the IOP line driver.

Description: Show that the IOP can drive both communication lines at the same time by showing overlapping of the T11 initialization and T11 completion of transmit messages.

Data elements collected: Entries into the KCT32 T11 dispatch table (interrupt vectors) for transmit enable and transmit complete were captured with the IRIG-B time. Thus, each entry into the DAS acquisition memory contained the dispatch table address and the IRIG-B time stamp.

Results.

Five hundred twelve transmit messages were captured containing the IRIG-B time stamp.

The I/O processor can drive both data communication lines at the same time. The transmit enable and transmit complete IRIG-B times overlapped by 1.1 seconds. With the total time for each transmission at 1.466 seconds we can safely say that the IOP software line driver can drive both lines at the same time.

Limitations/Special Considerations.

The DAS acquisition memory can hold 1024 elements of information which can be a very limited snapshot of activity. Unfortunately, the DAS cannot inform the host that the DAS trigger conditions have been met. If the tester wants to collect less than a full acquisition memory of elements, the tester must manually push the STOP switch on the DAS. Another consideration of data collection using this method is that the implementer must provide a host module to convert the data from the DAS into a suitable display format.

This MOP also demonstrated the use of an external time source to mark the time of occurrence of a monitored event. Although the central controller provides a time stamp on transfer of the acquisition memory, all events in the same buffer would be marked with the same time. An external time source can provide a more accurate time stamp for each event in a buffer, and also provides a means to synchronize the time used by multiple monitors.

Test No. 4.

Monitor type(s) demonstrated: SW

Description: Used to demonstrate a stand-alone software monitor capability, independent of any hardware monitor functions. MOP 8 was collected with this test.

Test Configuration.

Software monitor (MOP 8): The target software, resident on a VAX, was instrumented to collect transfers of control (procedure calls) of interest. For this test, the beginnings of all procedures and functions were instrumented. This instrumentation invoked a software kernel which obtained the procedure (function) name and passed this information to the software monitor, to be forwarded to the central controller for display and logging.

Hardware monitor: None.

MOP Category.

Primary: Software Functions

Secondary: Function Trace

Specific MOP.

Name: Software Function Trace.

Description: Trace of the invocation of software functions in the MCS application program.

Data elements collected: An ASCII string containing the traced subroutine/function name.

Results.

The kernel successfully passed back to the monitor a value which the monitor then converted to a procedure name which was displayed on the testers terminal and logged in a disk file.

The results clearly show a trace of subroutine and function invocations. With this information the control flow of the application can be easily followed.

Limitations/Special Considerations.

The software overhead of the kernel could cause a significant degradation of the application under test if available CPU resources are minimal or the monitored functions are extremely time critical.

Test No. 5.

Monitor type(s) demonstrated: HW

Description: Used to demonstrate the collection of MOP 10 via hardware monitoring techniques.

Test Configuration.

Software monitor: None.

Hardware monitor (MOP10): DAS 9100 with probes connected to the T11 processor of the KCT. The internal clock of the DAS was used to provide a measurement of the time interval between two events.

MOP Category.

Primary: Software Functions.

Secondary: Function Periodicity.

Specific MOP.

Name: KCT32 T11 polling periodicity

Description: Measurement of the period between polling operations of the T11 for IOP interrupts.

Data elements collected: A delta time element was collected from the DAS with a 100 ns resolution. The delta time is the measurement of the interval between successive polling functions of the T11.

Results.

The delta time was collected and displayed on the users terminal in real time at four second intervals. The delta time was also logged to a disk file.

The function periodicity was within specifications of the polling period.

Limitations/Special Considerations.

The hardware monitor can perform limited real-time analysis. For each delta time returned the DAS-VAX overhead was approximately 4 seconds. The majority of this time (about 3 secs.) was spent waiting on the DAS service request. The hardware monitor does not have the capability to compare a specified time with the measured time, which means that all events must be recorded and anomalous times detected during data reduction. A comparison capability could reduce the amount of data collected by recording only those events which exceed a given threshold.

Test No. 6.

Monitor type(s) demonstrated: SW.

Description: This test demonstrated the collection of multiple MOPs (11 and 12) with a single software monitor function.

Test Configuration.

Software monitor (MOPs 11 and 12): Two software kernels were used for this MOP. Both kernels were embedded within the application program. The kernels were designed to return time stamps to the monitor in a manner in which the monitor can calculate the function duration. The duration time was then passed to the central controller which displayed the data on the users terminal and logged the data into a disk file.

Hardware monitor: None.

Test No. 6 MOP No. 11

MOP Category.

Primary: Software Functions.

Secondary: Function Duration.

Specific MOP.

Name: Function duration of the MCS application transmit packet.

Description: Measurement of the duration of the MCS transmit packet function under various system loadings.

Data elements collected: The data elements collected were the beginning function time stamp and the ending function time stamp. Each time stamp had a resolution of 10 milliseconds.

Results.

The monitor collected the time stamps from the kernels at an interval of ten seconds. The monitor then calculated the function duration time and passed this value to the central collector.

The results showed that the function duration of the transmit message function fluctuated with system loading.

Limitations/Special Considerations.

A clock value must be available for the kernels to read so that they can pass the time values back to the monitor.

Test No. 6 MOP No. 12

MOP Category.

Primary: Software Functions.

Secondary: Function Processing Time.

Specific MOP.

Name: Processing time for the MCS applications function of transmission.

Description: Measurement of the processing time of the MCS transmit packet function under various system loadings.

Data elements collected: The data elements collected were the beginning function processing time and the ending function processing time. Resolution of the function processing time is 10 milliseconds.

Results.

The monitor collected the beginning and ending function processing time stamps from the kernels at an interval of 10 seconds. The monitor then calculated the function processing time and passed this value to the central controller.

The results showed that the function processing time of the transmit message function did not fluctuate with system loading.

Limitations/Special Consideration.

The embedded kernels must be capable of collecting CPU utilization data. If this element is not available then the implementer must have the knowledge and tools available to modify the operating system to log this information for kernel usage, or, alternatively, apply hardware monitoring methods.

Test No. 7.

Monitor type(s) demonstrated: SW, HW (Hybrid).

Description: Used to demonstrate a hybrid monitor configuration for MOPs 16 and 17. This test also demonstrated the collection of the same MOP (communication delay) with different monitoring techniques (software and hardware).

Test Configuration.

Software monitor (MOP 16): Two software kernels were used for this MOP. Both kernels were embedded within the application program. The kernels were designed to return time stamps to the monitor in a manner in which the monitor could calculate the communication delay. The communication delay was then passed to the central controller which displayed the data on the users terminal and logged the data into a disk file.

Hardware monitor (MOP 17): DAS 9100 with probes attached to the T11 processor of the KCT. The DAS internal clock was used to measure the interval between two events.

Test No. 7 MOP No. 16.

MOP Category.

Primary: Network Communication.

Secondary: Communication Delay.

Specific MOP.

Name: MCS application transmit packet communication delay.

Description: Measurement of the amount of time from when the application readies a packet for transmission to when the application receives an error free acknowledgement of the arrival of the packet.

Data elements collected: The data elements collected were the time stamps for the reading of the transmission packet and for the receiving of an acknowledgement. The resolution of the time stamps was 10 milliseconds.

Results.

The kernels successfully passed the time stamps to the monitor. The monitor collected this information from the kernels at ten second intervals and then calculated the communication delay time and passed this value to the central controller.

The results showed that the communication delay fluctuated with system loading.

Limitations/Special Considerations.

A clock value must be available for the kernels to read so that they can pass the time values back to the monitor.

Test No. 7 MOP No. 17

MOP Category.

Primary: Network Communication

Secondary: Communication Delay.

Specific MOP.

Name: KCT32 communication delay

Description: Measurement of the amount of time from when the KCT32 receives a transmit packet from the MCS application to when the KCT32 receives the completion of the acknowledgement.

Data elements collected: A delta time element was collected from the DAS with a 100 ns resolution. The delta time was a measurement of the period between the initialization of the transmission to the completion of the acknowledgement.

Results.

The delta time was collected and displayed on the users terminal in real-time at four second intervals. The delta time was also logged to a disk file.

The results showed that the communication delay fluctuated with systems loading.

Limitations/Special Considerations.

This test is adequate for statistical sampling or if the monitored event periodicity exceeds the DAS-VAX delay noted in prior tests.

Test No. 8.

Monitor type(s) demonstrated: HW

Description: Similar to MOP 6 of test 2 except that a time-of-day technique was used versus a delta time. Used to collect MOP 18.

Test Configuration.

Software monitor: None.

Hardware monitor (MOP 18): DAS 9100 connected to T11 processor of KCT and to IRIG time source.

Test No. 8. MOP No. 18.

MOP Category.

Primary: Component Usage

Secondary: Component Response (time-of-day)

Specific MOP.

Name: KCT32 T11 response time

Description: Measurement of the amount of time that the KCT32 takes for processing a transmit message.

Data elements collected: Entries into the KCT32 T11 dispatch table for transmit enable and transmit complete were captured along with the IRIG-B time. Thus, each entry into the DAS acquisition memory contained the dispatch table address and the IRIG-B time stamp. The resolution of the IRIG-B was 100 nanoseconds.

Results.

Five hundred twelve (512) transmit message events and associated IRIG-B time stamp were collected.

Component response does vary with buffer lengths, traffic loads, and data transfer lengths.

Limitations/Special Considerations.

See test 3, MOP 2.

APPENDIX H
DISTRIBUTION

DISTRIBUTION LIST

<u>Addressee</u>	<u>Number of Copies</u>
Commander U.S. Army Test and Evaluation Command ATTN: AMSTE-TC-M	3
AMSTE-TO	2
AMSTE-EV-S	1
AMSTE-TE	6
Aberdeen Proving Ground, MD 21005-5055	
Commander Defense Technical Information Center ATTN: DTIC-DDR	2
Cameron Station Alexandria, VA 22314-5000	
Commander U.S. Army Combat Systems Test Activity ATTN: STECS-DA-M	2
Aberdeen Proving Ground, MD 21005-5000	
Commander U.S. Army Yuma Proving Ground ATTN: STEYP-MSA	2
Yuma, AZ 85364-5000	
Commander U.S. Army Jefferson Proving Ground ATTN: STEJP-TD-E	1
Madison, IN 47250-5000	
Commander U.S. Army Dugway Proving Ground ATTN: STEDP-PO-P	1
Dugway, UT 84022-5000	
Commander U.S. Army Cold Regions Test Center ATTN: STECR-TM	1
APC Seattle, WA 98733-5000	
Commander U.S. Army Electronic Proving Ground ATTN: STEEP-MO-M	4
Fort Huachuca, AZ 85613-7110	

Addressee

Number
of Copies

Commander
U.S. Army Tropic Test Center
ATTN: STETC-TD-AB
APO Miami, FL 34004-5000

1

Commander
U.S. Army White Sands Missile Range
ATTN: STEWS-TE-PY
 STEWS-TE-O
 STEWS-TE-M
 STEWS-TE-A
White Sands Missile Range, NM 88002-5000

4

1

1

1

END

6-87

DTIC