

DTIC FILE COPY

DC-TR-90-3066

2

# AD-A229 800

TECHNOLOGY FOR DEVELOPMENT AND VERIFICATION  
FLIGHT CRITICAL SYSTEMS



De Feo/D. Mann

PARTE, Inc.  
3041 Avenida de la Carlota  
Suite 400  
Ft. Lauderdale, FL 33309

October 1990

Final Report for Period Jan 90 - Jun 90

Approved for public release; distribution unlimited

DTIC  
ELECTE  
DECO 4 1990  
S B D  
Co

LIGHT DYNAMICS LABORATORY  
WRIGHT RESEARCH DEVELOPMENT CENTER  
AIR FORCE SYSTEMS COMMAND  
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6553

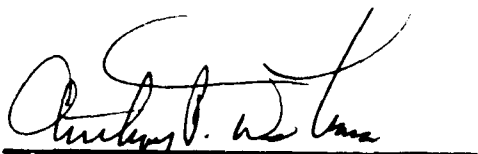
DTIC FILE COPY

NOTICE

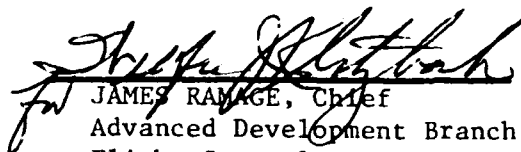
When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.



ANTHONY P. DeTHOMAS  
Project Engineer  
Advanced Development Branch  
Flight Control Division



JAMES RAMAGE, Chief  
Advanced Development Branch  
Flight Control Division

FOR THE COMMANDER



H. MAX DAVIS, Assistant for  
Research and Technology  
Flight Control Division  
Flight Dynamics Laboratory

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/FIGX, WPAFB, OH 45433-6553 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS None			
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited			
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-3066			
6a. NAME OF PERFORMING ORGANIZATION SPARTA, Inc.		6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Flight Dynamics Laboratory (WRDC/FIGX) Wright Research Development Center		
6c. ADDRESS (City, State, and ZIP Code) 23041 Avenida de la Carlota, Suite 400 Laguna Hills, CA 92653-1507		7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6553			
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as 7a		8b. OFFICE SYMBOL (If applicable) FIGX	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-89-C-3608		
8c. ADDRESS (City, State, and ZIP Code) Same as 7b		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO. 65502F	PROJECT NO. 3005	TASK NO. 40	WORK UNIT ACCESSION NO. 52
11. TITLE (Include Security Classification) Methodology for Development and Verification of Flight Critical Systems					
12. PERSONAL AUTHOR(S) P. De Feo/D. Mann					
13a. TYPE OF REPORT Final		13b. TIME COVERED FROM Jan 1990 TO Jun 1990	14. DATE OF REPORT (Year, Month, Day) 1990, October		15. PAGE COUNT 33
16. SUPPLEMENTARY NOTATION This is a Small Business Innovative Research Program, Phase I report.					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	VMS, Transputers, Architecture Evaluation, Preliminary Design, Emulation		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This project addresses the technology needs of Vehicles Management Systems (VMS) during the critical phases of architecture definition and preliminary design. VMS are flight systems which are highly integrated for enhancing maneuvering performance, weapon delivery and fault tolerance. SPARTA has developed and demonstrated a powerful and flexible rapid prototyping environment, Transputer base Advanced Development Environment or TRADE, where VMS architectures can be rapidly emulated and evaluated. The environment includes a SUN workstation and a network of transputers. Transputers are single chip computers which provide extensive processing and communication capabilities at very low cost. The environment enables the system designer, via a graphic based User Interface (UI), to: 1) rapidly and easily modify the VMS architecture as well as the structure of the software embedded in each VMS processor and 2) control the entire environment. Evaluation tools which address system level issues are also included in the environment, so that the relative merits of competing architectures can be quantitatively evaluated.					
20. DISTRIBUTION / AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. A. DeThomas			22b. TELEPHONE (Include Area Code) (513) 255-8474	22c. OFFICE SYMBOL WRDC/FIGX	

METHODOLOGY FOR DEVELOPMENT AND VERIFICATION OF  
FLIGHT CRITICAL SYSTEMS.

PHASE 1 FINAL REPORT

TABLE OF CONTENTS

	Page
1.0 PROJECT SUMMARY	2
2.0 STATEMENT OF NEEDS	2
3.0 PROGRAM OBJECTIVES	3
4.0 PHASE 1 OBJECTIVES	5
5.0 PHASE 1 ACTIVITIES AND RESULTS	6
5.1 TRADE HARDWARE CONFIGURATION	7
5.2 TRADE SOFTWARE CONFIGURATION	9
5.3 TRADE SIMULATION CAPABILITIES	10
5.3.1 BUS SIMULATION	10
5.3.2 FLIGHT SOFTWARE SIMULATION	11
5.3.3 COMMUNICATION AND SYNCHRONIZATION	12
5.3.4 EVALUATION TOOLS	13
5.3.4.1 VARIABLE LATENCY TIMES	14
5.3.4.2 BUS UTILIZATION MONITOR	16
5.4 USER INTERFACE	16
6.0 CONCLUSIONS	19



by _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

## 1.0 PROJECT SUMMARY

This project addresses the technology needs of Vehicle Management Systems (VMS) during the critical phases of architecture definition and preliminary design. VMS are flight systems which are highly integrated for enhancing maneuvering performance, weapon delivery and fault tolerance. SPARTA has developed and demonstrated a powerful and flexible rapid prototyping environment, Transputer base Advanced Development Environment or TRADE, where VMS architectures can be rapidly emulated and evaluated. The environment includes a SUN workstation and a network of transputers. Transputers are single chip computers which provide extensive processing and communication capabilities at very low cost. The environment enables the system designer, via a graphic based User Interface (UI), to: 1) rapidly and easily modify the VMS architecture as well as the structure of the software embedded in each VMS processors and 2) control the entire environment. Evaluation tools which address system level issues are also included in the environment, so that the relative merits of competing architectures can be quantitatively evaluated.

## 2.0 STATEMENT OF NEED

Vehicle Management Systems (VMS) integrate flight control functions, propulsion control functions and utility functions, to achieve enhanced performance and mission and safety reliability. VMS architectures are very complex, highly distributed, and perform tasks with different levels of criticality. A generic VMS architecture is shown in Figure 1. DoD and NASA have sponsored several programs for identifying promising VMS architecture configurations. Among them: 1) Design Methods for Integrated Control Systems (DMICS), sponsored by the Air Force, 2) Highly Integrated Digital Electronic Control (HIDEC), sponsored by NASA, and 3) Integrated Airframe/Propulsion Control System Architecture (IAPSA) also sponsored by NASA. These programs, and other programs funded with IR&D money, have all identified critical technology needs in the areas of design and evaluation of VMS architectures. A most critical technology need exists in the area of tools and techniques for supporting the early system development phases, including requirements definition, specifications and

preliminary design. These tools are needed for establishing feasibility, defining the specifications, and quantitatively evaluating the relative merits of competing architectures, prior to starting the detail design work. Examples of the critical Figures Of Merit (FOM) of alternate configurations include: 1) the distribution of the overall computational load among the VMS processors; load distributions which result into even duty cycle for each processors are one important design goal, 2) the complexity of the architecture as is reflected, for instance, by the interprocessors data communication requirements; architectures which minimize data distribution requirements and evenly distribute those requirements among all available resources, can increase the operational reliability of the system, reduce the life cycle cost, and can support algorithms with high dynamics content, 3) the level of partitioning which is provided among functions with different levels of criticality and the validation of partitioning, 4) Failure Detection, Isolation and Reconfiguration (FDIR) issues like the fault coverage, the time required for failure isolation and reconfiguration and the reconfiguration capabilities, 5) the statistical distributions of computational delays and transport lags within the network; uniform distributions result in predictable system dynamic performance.

The technology currently available for supporting those tasks is rather limited, at best. The same technology needs are not limited to digital flight systems but they are common to many military and commercial applications which require highly distributed processing architecture for performing critical real time tasks. Examples of such applications are the control of nuclear reactors, integrated Guidance, Navigation & Control of space vehicles including the Space Station, and many others.

### **3.0 PROGRAM OBJECTIVES**

SPARTA approach for addressing the critical technology needs previously described is to develop a Transputer based Advanced Development Environment (TRADE) where VMS architectures can be easily and rapidly prototyped and critical performance parameters and FOMs can be credibly and quantitatively evaluated. SPARTA has developed a core implementation of TRADE, Phase I TRADE, for feasibility demonstration purposes. Phase I TRADE is discussed in detail in the following sections of this report. In the final

configuration TRADE will include a workstation linked to a network of transputers. A transputer is an advanced single chip computer which provides extensive processing and communication capabilities at very low cost. Transputer technology makes it cost effective to emulate complex architectures by mapping each embedded processor into one dedicated transputer. Within the TRADE environment shared resources, like communication busses, will also be emulated in dedicated transputers. Critical characteristics of the embedded flight software will be simulated in each transputer including the real time software structure, I/O processes and synchronization algorithms. The software simulation could be step-wise refined as the VMS software design evolves. TRADE will use off-the-shelf software development environment targeted to FORTRAN, C and ADA, as the embedded languages, so that the effects of specific language constructs, like Ada tasking in the case of ADA, on synchronization and timing can be properly evaluated.

TRADE will include a variety of tools which support the evaluation process of the VMS architecture including tools for: 1) evaluating duty cycles of processors and communication links, 2) demonstrating partitioning, 3) measuring the time to detect failures and to reconfigure, 4) measuring statistical distributions of time lags and transport delays. TRADE will have the hooks and scars for hosting additional evaluation tools as needed. The tools will be hosted in the SUN workstation and will require some instrumentation of the transputers software.

An important element of TRADE will be an advanced User Interface which will be implemented in the host workstation and which will provide the VMS designer with a Macintosh like graphical interface. The User Interface will support: 1) arranging the transputers in a variety of architectures, including redundant architectures, so that the user will not be required to change or rearrange computer boards every time a new target architecture must be emulated; 2) simulating a variety of real time software structures by simply selecting, from a menu of options, different structures and/or parameters 3) emulating a variety of bus protocols from a library of available protocols. Additional protocols could be easily added as required; 4) controlling and managing all aspects of the evaluation environment including selecting the evaluation tools, the set of test data, the duration of the test and the display of the results.

All capabilities provided by the TRADE environment are generic in nature, and therefore they are broadly applicable to any application which requires distributed computing architecture for performing time critical functions. An example of such applications is industrial process control, like the control of nuclear reactors, continuous industrial process such as paper making, metal/plastic, rubber sheet forming, flight systems for commercial aircraft, and space systems.

#### **4.0 PHASE 1 OBJECTIVES**

The objectives of Phase 1 of this program are:

- 1) To establish the feasibility of the approach. SPARTA successfully established the feasibility of the approach by developing and demonstrating key critical features of TRADE.
- 2) To identify solutions to all issues which could not be directly addressed under the limited Phase 1 effort. SPARTA has identified promising solutions to all critical issues which could not be demonstrated in this phase of the program because of the limited available resources.
- 3) To formulate the Phase II plan including cost estimates and schedule. SPARTA has defined in detail a development plan for TRADE which is cost effective to the Government and presents no technical risks. The plan effectively draws from the results of Phase I of the program.

In summary all Phase I objectives have been achieved. Specifically, the feasibility of the proposed approach was demonstrated by actually implementing most of the technologically advanced aspects of TRADE rather than simply performing a paper study. SPARTA is confident that a Phase II program will be very beneficial to the Government and can be carried out with minimum risks. SPARTA is committed to support this program with internal funds beyond the Phase II so that a versatile product for military and commercial applications can be developed and marketed.



## 5.0 PHASE I ACTIVITIES AND RESULTS

SPARTA has developed Phase I TRADE which includes critical elements of TRADE. Phase I TRADE has been implemented using transputer technology, for the purpose of demonstrating the feasibility of the proposed approach. SPARTA have combined the flexibility of a SUN workstation interface with the distributed processing power of a transputer network to provide a dynamically configurable, real time hardware simulation environment of VMS architectures. A functional diagram of the system is shown in Figure 2.

The SUN performs the functions of Host Computer for the graphic based user interface, the embedded software development environment and the evaluation tools. The user has full control of the TRADE environment from the SUN.

Transputers are a product of INMOS, a British company with a large representation in the US. Several US and foreign system houses have used INMOS transputer chips for building single board processors and providing integrated computing and communication environments. The technology, however, as previously stated, is state of the art and even integrated products require significant software development to provide basic services. TRADE use a transputer environment developed by Topologix, a US system house, which includes four INMOS transputer chips (T800 model) in a single board and which provides an integrated software development environment. SPARTA does not intend to use Topologix technology in the Phase II of this program, as further discussed later in this report.

Within the TRADE network one transputer processor (T800) performs the function of embedded bus simulator. It controls the network topology and establishes the baud rate in response to user commands from the SUN station. A "1553 like" bus protocol is currently implemented in TRADE. The remaining T800 transputers emulate embedded processors within the VMS architecture. TRADE provides the user with the capability of rapidly prototyping the structure of the real time software hosted in each transputer. Specifically, multirate real time execution structures can be achieved by: 1) implementation of executive logic trees, which direct execution to

alternate paths based on the required frequency of execution, or 2) by a table structure where each entry in the table represents a module to be executed and all entries are sequentially executed. The user can easily select either structure and all required parameters by using the handles provided by the user interface. TRADE transputers, like the embedded processors in VMS architectures, can execute synchronously or asynchronously with respect to each other. Synchronization within the Transputer network is I/O driven, as it is further elaborated later in this document.

All TRADE configuration parameters can be easily selected by the user which only interacts with the SUN workstation via a "Macintosh like" graphical interface which was implemented utilizing X-Windows software.

During the set-up phase, the user configures the transputer network by utilizing the handles provided by the User Interface. User commands are routed from the SUN to the transputer network which automatically configures itself in response to the user's commands. Once that the transputer network is properly configured, the user initiates real time execution phase by pressing the start button on the SUN set-up screen. The simulation stops automatically after an user selected amount of time has elapsed. Simulation results, like bus utilization and timing distributions, are periodically updated on the control panel as the execution progresses.

TRADE is currently implemented in a distributed workstation environment as shown in Figure 3. The transputers resides in the VME chassis of a SUN 3/160 workstation which is connected to the rest of the network through an Ethernet Bus. The distributed network approach allows for the development and operation of the simulation from any computer on the Ethernet. The X-Windows based User Interface is transportable to the VAX and therefore TRADE can be controlled from that workstation.

## **5.1 TRADE hardware configuration**

TRADE is currently implemented with 4 INMOS T800 transputers installed on a Topologix T1000 mother board which is connected to a SUN 3/160 through a VME Bus. Each T800 processor is a complete computer which includes CPU, floating point unit (FPU), 4

communication channels (interlinks) and 4 KB of memory within one VLSI package. 4 Mbytes of external memory is added to each transputer via the memory interface. The architecture of the T800 is shown in Figure 4. SPARTA ran several benchmark programs for the CPU and the FPU and clocked the processors at 5 MIPS and 600 KFLOPS respectively, which is less than the nominal speed.

The unique features of the T800 processors are the 4 interlinks. These direct memory access (DMA) channels operate asynchronously, in parallel, at very high rate (20 Mb). Communications on one channel do not significantly impact either the CPU or any other channels. Interlinks are dual ported and provide point to point communications. The most common use is that of connecting pairs of T800. They can also connect a T800 to a serial latch chip (IMS CO11/12), another INMOS product, which in turn can interface with such external devices as Analog to Digital (A/D) converters, Digital to Analog (D/A) converters and others.

The four TRADE T800 processors are tied together on a Topologix T1000 VME board (Figure 5). The board can be connected to numerous other T1000 boards to form a large network capable of emulating even the most complex VMS architectures. Transputer pairs can be directly interlinked, as previously discussed. The key to the expansion capability of the network is, however, the C004 Link Crossbar Switch, another INMOS product, which can dynamically interlink T800 pairs or link T800 to other devices like the CO11/12 link adaptor. The current TRADE configuration includes the crossbar switch but not the link adaptor. The CO11/12, or equivalent link adaptors, will be considered for inclusion in the TRADE environment during Phase II of this effort, to provide the flexibility of interfacing TRADE to external equipment. The current TRADE configuration does include, however, the C004 programmable switch which is controlled by the T212 network controller which in turns is controlled by the operating system on the SUN. The T212 can only be programmed during the booting process. As a result it is necessary to reboot the T1000 every time that a new VMS configuration must be evaluated. This limitation will be removed during the Phase II of this program because TRADE will utilize a more advanced transputer system than that currently used.

The Topologix T1000 board is installed on the VME bus of a SUN 3/160 workstation. Only one T800 (transputer A in Fig. 5)

interfaces directly to the VME bus. The other T800s must go through processor A to access that bus. This limitation will also be removed during Phase II of this program.

## **5.2 TRADE software configuration**

In the TRADE environment, the SUN workstation communicates to the T800 transputers through LogixOS, an operating system developed by the Topologic Corporation. LogixOS is a UNIX like real time operating system which is stored in each T800 and supports parallel processing. LogixOS is integrated with the UNIX environment on the SUN workstation and thus it provides an integrated development environment for the T800 software which includes a T800 compiler, a rather primitive debugger, booting, memory allocations, etc. LogixOS also provides some basic real time services like "read" and "write". One significant short coming of the LogixOS package is the lack of a source level debugger which controls execution of the T800 programs and monitors the communications and messages among processors. The lack of such debugger made the debugging of the TRADE software a very laborious task which was only supported by a primitive break point insertion capability. SPARTA is convinced that the software development environment for Phase II of this program must include a source level debugger.

Communication between TRADE processors is performed utilizing the LogixOS services previously described. LogixOS provides bus level and network level communications.

Bus level communications have minimum overhead; data is sent down a DMA channel without regard for which processor is connected at the other end. The programmer, however, is required to implement the necessary low level functions including controlling the routing between the communicating processors and managing the handshaking between sender and receiver. Message routing is not a simple task in the case of complex network and it may require to relay data through several processors. Also The sending processor cannot continue (will block) until the receiving processor takes the incoming data. While bus level communications requires some programming overhead, the speed makes it a candidate for communication in Phase II.

TRADE utilizes network level communications. With this method, communications messages are routed from processor to processor (rather than data channels) until the destination processor is reached. Background mail daemons in each T800 determine the optimal communication path upon receipt of the data and the attached destination address. Network level communication is very easy to program. The system overhead, however, is much larger than for bus level communications. LogixOS has a high overhead (600.  $\mu$ s) because it does not build a topology data base and because it implements the system calls in "C". Other operating systems, like Express, have much smaller overhead (100  $\mu$ sec.) because they use a topology data base for determining the best routing and implement the calls in assembly.

### **5.3 TRADE simulation architecture**

TRADE simulation environment includes: 1) VMS bus, 2) embedded flight software, 3) synchronization and communications, 4) evaluation tools and 5) user interface. Each component is further discussed

#### **5.3.1 Bus simulation.**

In TRADE we have implemented a bus environment which simulates the functions of the Remote Terminal (RT), and Bus Controller (BC) of the 1553 bus protocol. The BC, and the actual bus, are simulated in the T800 bus simulator. All other T800s are simulated as RTs. Data transmission starts with one RT processor requesting the BC for access to the bus. If the bus is free a confirmation message is sent to the RT and data is transferred from the originating RT processor, through the bus simulator, to the destination processor, which is another RT. If the bus is busy, a denial message is sent to the RT which waits for a user adjustable interval of time and then makes another bus request. The BC calculates the message transmission time based on the length of data to be transferred and the bus baud rate. The receiving TR gains access to the data after the transmission time has elapsed. The BC also tracks and reports bus utilization data.

The bus structure is specified by the user from a file in a table format as shown in Figure 6. The first column in the table represents the bus identifier number. Each remaining column represents a unique processor which can be either the bus BC (-1 designator), an RT (1 designator) or not connected to the bus (0 designator). In this example a pipeline structure is specified. There are two buses, 0 and 1. Bus 0 connects processors 1 and 2, while bus 1 connects processors 2 and 3. With this method bus topologies can be rapidly changed by loading a new bus specification file. The number of processors required to simulate a bus depends on the number of processors connected to the simulated bus.

### 5.3.2 Simulation of embedded flight software.

TRADE provides the capability of easily and rapidly prototyping flight software embedded in each processor. The structure of the programs, execution times, parameter updates, and communication requests are specified in a processor format file (PFF). There is one PFF per each processor. An example of PFF, for processor 1, is shown in Figure 7. The software includes 6 tasks, task 0 through 5 (column 1). The table shows the order of execution of the tasks (0, 1, 3, 0, etc.). After the last task (task 5) of the sequence is executed, then control is transferred back to the first task of the sequence (task 0) and the entire sequence is executed again. The process is repeated until the user interrupts the execution. Task 0 is executed four times in the sequence, the highest execution rates of all tasks, because it includes the processor 1 functions with the highest dynamic content. Tasks 1 and 2 are executed twice; their algorithms have less dynamic content than those of task 0. Tasks 3, 4 and 5 are executed only once in the sequence. They have the lowest dynamic content. Task 5 represents low priority, interruptable, background activities, with no real time consequences. In column 2 the number of variables which are updated in each tasks are shown. Tasks 1, 2 and 3 update one variable each; tasks 0, 4 and 5 do not update any variables. The estimate of the execution time of each task in the target processors are shown in column 3. The execution time of task 0 is estimated to be 5 msec. In column 4 the destination processor is shown for the data updated in processor 1 (a negative number means that the data is not sent to any processors). The ID of the bus used for data transmission is shown in column 5. Finally in the last column the source processor is shown for data which may be

required by the tasks executing in processor 1. A negative value indicates that no data is needed.

The PFFs drive the execution sequence in each T800 processor. When the simulation is initialized, each processor reads its PFF. At the start of a new task, each processor first checks if data is to be sent. If so, a data message is formed and a bus request is made to the BC. Program execution in that T800 stops until the bus is available. Next the processor receives data if so specified in the PFF. The mechanism for each T800 to get access to data from other processors is explained later. Finally, the program enters a wait loop of duration equal to the estimated execution time in the PFF for that task. In Phase II of this program the facility of simulating the embedded flight software will be provided.

Each parameter that is generated is marked with a global time stamp. This stamp is passed with the data between processors and is used to measure the age of the data. During the instruction step when data has been received the age of the variables are recorded. These ages or delays are transformed into a histogram and passes to the host computer for user evaluation.

The PFF structure implemented in TRADE is a powerful tool for rapid prototyping the execution sequence of the flight software. In Phase II the PFF structure will be enhanced so that different tables can be made "active" for each processor based on the status of logic variables. Then the structure of the software executing in each processor can be changed in real time. This allows for simulating: a) the different execution paths in the flight processors due to flight conditions and control mode switching, and b) software reconfiguration due to failures.

### **5.3.3 Interprocessors communications and synchronization.**

Interprocessors communication are supported by the "mail daemon" a LogixOS background process hosted in all T800 processors. When one processor "send" data to another processor, the data is initially stored in the mail daemon First-In-First-OUT (FIFO) buffer of the receiving processor. "Send" is an "unblocked" instruction because program execution resumes immediately after execution of "send". In the receiving processor, the data is transferred from the FIFO

buffer to memory by executing a "receive" instruction. "Receive" is a "blocked" instruction because program execution can only resume after the data is actually available in the FIFO buffer.

TRADE utilizes the data exchange facility of LogixOS for synchronizing pairs of processors. For this, in both processors a "send" instruction is immediately followed by a "receive" instruction (Figure 8). Processor A executes the "send data to B" instruction. Next A instruction is a "receive data from B". Because the data from B is not available yet, A puts itself into a waiting loop until that data is available. Finally processor B executes the "send data to A" instruction and immediately afterwards executes the next "receive data from A" instruction. At the same time A can also execute the "receive data from B" instruction, and synchronization is achieved. A limitation of this approach is that it works for only 2 processors. In Phase II, SPARTA will develop a synchronization technique based on unblocked broadcast instructions which will provide a mean of synchronizing more than two processors. The technique will be used to simulate broadcast interrupts issued during the background task.

During Phase II of this program SPARTA intends to develop and evaluate another approach for synchronizing T800 processors based on time interrupts issued by the BC which is hosted in the bus simulator.

#### **5.3.4 TRADE evaluation tools.**

In its final configuration TRADE will include a variety of tools to primarily address architectural and design issues. The tools will be *hosted in the SUN workstation and may require special instrumentation of the simulated target software.* It is anticipated that TRADE will effectively support a variety of applications which require distributed processing architectures. TRADE will need then to be tailored to reflect the specific requirements of the products supported and those of the user development environment. In any case, TRADE is not expected to be a static environment. On the contrary, as the user becomes more familiar with the capability of TRADE, he will most likely discover the need of special tools that he wishes to have integrated within the TRADE environment. All this leads to the need for TRADE to provide an environment which can be easily modified and new tools accommodated. This is one of the



reasons why SPARTA decided to use a SUN workstation as the host computer of TRADE. The SUN workstation with a VME based transputer system provides the greatest flexibility for future expansion and product support.

During Phase I of this program SPARTA has developed, and integrated in the TRADE environment, two tools for evaluating critical figures of merits of the target architectures. The tools are clearly not in their final version, and still much work needs to be done to improve their performance and ease of use. The tools, however, demonstrate the feasibility of the approach because they required the implementation of some critical functional capabilities.

#### **5.3.4.1 Variable latency.**

This tool computes the latency of user selected variables as the difference between the time the variables are used in one processor and the time the variables were last updated in another processor. Latency time of each variable varies every simulation execution cycle as a function of many factors including bus utilization and computational skewness between the processors (the time granularity of the T800 is 64.  $\mu$ sec.). As a result, at the end of a simulation run, the latency time of each variable is an histogram, rather than a single value. Of particular relevance to VMS configuration is the definition of the highest latency values during a simulation run. High values are not desirable because they decrease the stability of control loops and limit the dynamic content of the embedded algorithms.

The implementation of this tool requires the establishment of a global time which is common to all processors, so that the time tagging process is independent from the processor where tagging takes place. Each processor has its own clock, and therefore, its own time. Time zero within a processor is defined by LogixOS when the node is booted. The nodes are booted from the host in a serial fashion which results in time offsets among the processors equal to the time it takes to boot a node (Figure 9). For the Topologix operating system (used in Phase I) this offset is on the order of 20 seconds; other Operating Systems, like Express, require much less time to boot (about 1 sec) and generate correspondingly smaller offsets. In any cases, the time offset among all processor pairs

must be either eliminated or accurately determined so that the effects can be properly biased out.

In the TRADE environment the clock time of one processor, called reference processor, is arbitrarily chosen to represent the network global time. All other processors estimate their offset from that global time. The process of determining the time offsets is called "time registration" which is a timed handshake process. The time offset is estimated by timing the response of a remote processor to a "send time" request from the reference processor (Figure 10). The reference processor first records its own global time ( $t_{G1}$ ) and immediately requests the local time from a remote processor. As soon as the remote processor receives the request, it records its own time ( $t_{L2}$ ) and returns a copy of it to the reference processor. Finally the reference processor records the time that the response is received ( $t_{G3}$ ). The reference processor can then estimate in global time units when the remote processor recorded its own time ( $t_{G2}$ ):

$$t_{G2} = (t_{G3} + t_{G1})/2$$

The time bias ( $t_b$ ) between the two processors can then be estimated by:

$$t_b = t_{L2} - t_{G2}$$

The same process is repeated throughout the network so that each processor has an estimate of its own offset with respect to the global time. This process proved to be very accurate and repeatable among all combinations of the 4 available processors. The handshake time ( $t_{G2}$ ) was determined to be equal to 1.3 ms in all cases.

During the Phase II of this program an alternate method will be implemented and evaluated for synchronizing the clocks. In this method all clocks will be initialized to zero when a global interrupt is given. An interrupt timing scheme poses a communication challenge in a mesh network environment, like that of TRADE. In fact, transputers utilize dual port communications which prevent the implementation of broadcast interrupt which are simultaneous to all processors (Figure 11). In the TRADE environment a cascaded

interrupt is utilized to broadcast a message. An arbitrary processor is chosen to start the process. It sends the interrupt to its immediate neighbors which then record the local times and immediately send another interrupt to their neighbors, and so on until all processors have been interrupted. The time offsets among processors is expected to be negligible and certainly well within the clock granularity. The relative merits and weaknesses of the two methods, the time handshake and the cascaded interrupt, will be evaluated during Phase II of this program.

#### **5.3.4.2 Bus utilization.**

In VMS configurations common resources like communication busses are shared by many processors which typically perform functions with vastly different dynamic contents and levels of criticality. Architectures which require high frequency interprocessors exchanges of large volume of data put high demand on the bus bandwidth. This is not desirable because high bandwidth utilization increases the system failure rate, increases the life cycle cost, induces instability in high dynamics control algorithms, and reduces the capability of system growth. Bus bandwidth utilization is therefore a critical FOM of VMS architecture which must be properly estimated and evaluated. Bus bandwidth utilization is not constant with time but varies as a function of flight control modes, flight conditions, and more generally, as a function of the program paths executed each frame. It is important to determine the statistical distribution of bandwidth utilization so that the conditions of peak demands are identified and evaluated.

TRADE includes a core implementation of a tool for identifying and displaying bus utilization. Bus utilization is measured by the BC. A percent utilization is displayed for each bus in the users control panel which will be described in the next section. Bus utilization is defined as the ratio between the time data is actually transferred through the simulated bus, to the total run time of the simulation.

#### **5.4 User Interface (UI).**

The TRADE user can manage and control the entire simulation environment through a graphical interface running on a SUN workstation. The SUN is part of an Ethernet based distributed

workstation environment. The UI program is written using X-Windows so that it provides a "Macintosh like" look and feel. Any workstation in the network can remotely operate the simulation. Furthermore, since X-Windows is an open standard, the simulation can easily be ported to other workstations such as VAX, Hewlett-Packard, or Apollo. The current TRADE UI is rather basic and provides limited capabilities only. It does demonstrate, however, the feasibility of the approach and gives a feel of the capabilities which can be implemented using the powerful graphic environment of X-Windows. TRADE UI will be much enhanced during the Phase II of this program.

The UI includes two panels: the control panel and the display panel (Figure 12). The typical user course of action includes: a) simulation of the architecture, b) simulation of the embedded software structure, c) invocation of the evaluation tools, d) running the simulation, and e) displaying the results. Each action is discussed further in the following paragraphs.

**Simulation of the architecture.** The User selects the bus topology among the options provided under the "Processor topology" heading of the control. TRADE currently includes two topologies: pipeline and shared bus. These two topologies are selected by clicking the mouse in the appropriate check boxes. The chosen topology is graphically displayed in the topology window. In this example, the shared bus structure has been chosen.

**Simulation of the embedded software structure.** The user specifies the embedded program architecture by creating PFFs, one for each T800 in the network. The current UI allows the user to preset PFFs and to group them into directories. Each directory includes four prespecified PFFs, one for each of the four T800 in TRADE. The user can select, edit, and create directories and PFFs by pressing the "Load Processor" button in the control panel. After properly configuring the PFFs, the user can load the software in the respective processors by simply clicking on the "LOAD" button (not shown in Figure 12). At this point, the entire simulation has been configured and is ready to run.

**Invocation of the evaluation tools.** The user can select: a) which variables to monitor relative to time latency, b) which bus to monitor relative to bus utilization.

The user is able to tailor the simulation with three groups of controls. The first group, "variable selection", allows the user to select which parameter to track during the simulation. The generating processor ID is entered for the node number and the parameter ID number is entered for the variable number. These values are placed in a global register for use by the embedded programs during operation.

The second group, "histogram controls", allows the user to control the histogram of time latencies of critical variables. The maximum and minimum delay values of the histogram are set with two slide switches. Because the histogram has a fixed number of bars, the minimum and maximum values also sets the resolution of the histogram. This method allows the user to pan and zoom the time delay of the variable of interest.

The third group, "system overhead", allows the user to insert two overhead values into the simulation: the process and the communication overhead. The process overhead is a delay value added to each of the processor tasks defined in the process specification table. This allows the user to simulate system overhead induced by the real time executive hosted in each TRADE processors. The communication overhead is a delay inserted at the beginning of each message passed between processors. This allows the user to simulate the communication overhead generated by the the bus protocol.

**Running the simulation.** After all simulation parameters have been properly set the user can initiate the execution of the simulation. Execution is initiated and terminated by toggling the start/stop button in the right center of the control panel. During the simulation, the current frame number of processor 1 is displayed in the upper right hand corner of the display panel. By default, the simulation will stop after 1000 frames have been executed.

**Displaying the results.** The average bus utilization during the entire simulation run is computed by the BC. Bus utilization and the histogram of variable latency time are passed to the UI program after the simulation execution is halted. The bus utilization is displayed as a percentage in the upper right hand corner of the

control panel. The delay histogram is plotted in the lower right hand display window.

## 6.0 CONCLUSIONS

All the key objectives of the Phase I program have been achieved and successfully demonstrated. Specifically we have demonstrated that it is feasible and cost effective to develop TRADE, a rapid prototyping facility of VMS configurations by utilizing transputer technology. SPARTA has achieved the objectives by developing Phase I TRADE which integrates many of the critical requirements of that environment. Phase I TRADE is just a core implementation of that environment. During Phase II of this program SPARTA plans to develop a prototype version of TRADE, PHASE II TRADE, and during Phase III of this program, which will be internally funded, a production unit of TRADE will be developed. Phase II TRADE will use a set of hardware and software different from that of Phase I TRADE. Phase I TRADE utilizes resources available at SPARTA at no cost to the Government. The full set of capabilities which will be included in the final product will be developed and integrated in Phase II TRADE. Phase II TRADE will include a fully validated Ada software development environment for simulating the target software. It will also include a fully developed simulation capability, analysis tools including code instrumentation facility and fault insertion mechanisms, and a graphic oriented user interface. Phase II TRADE will be delivered to the Government and will be configured to emulate a VMS configuration which will be jointly selected by the Government and SPARTA. The attached Phase II proposal further elaborates on the Phase II objectives.

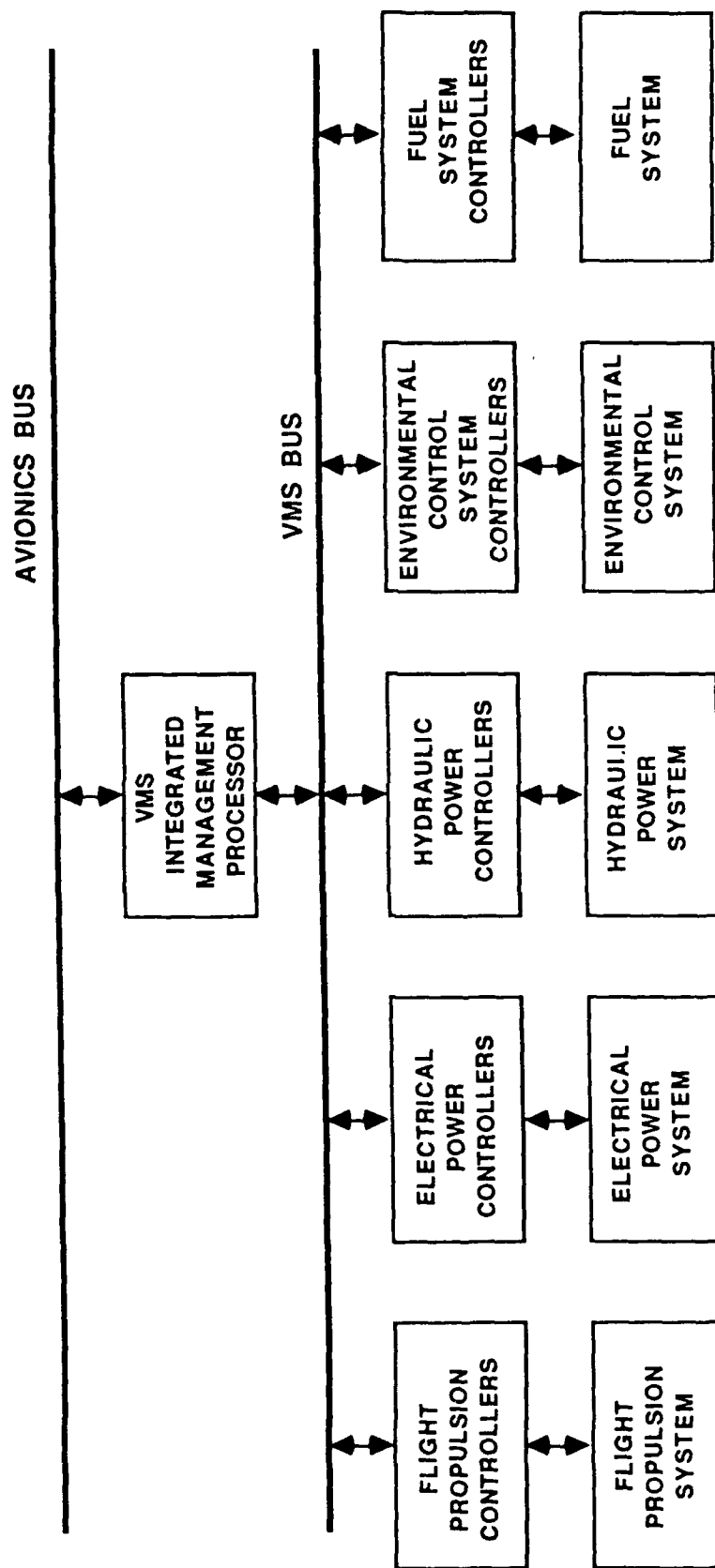


FIGURE 1. GENERIC VMS ARCHITECTURE

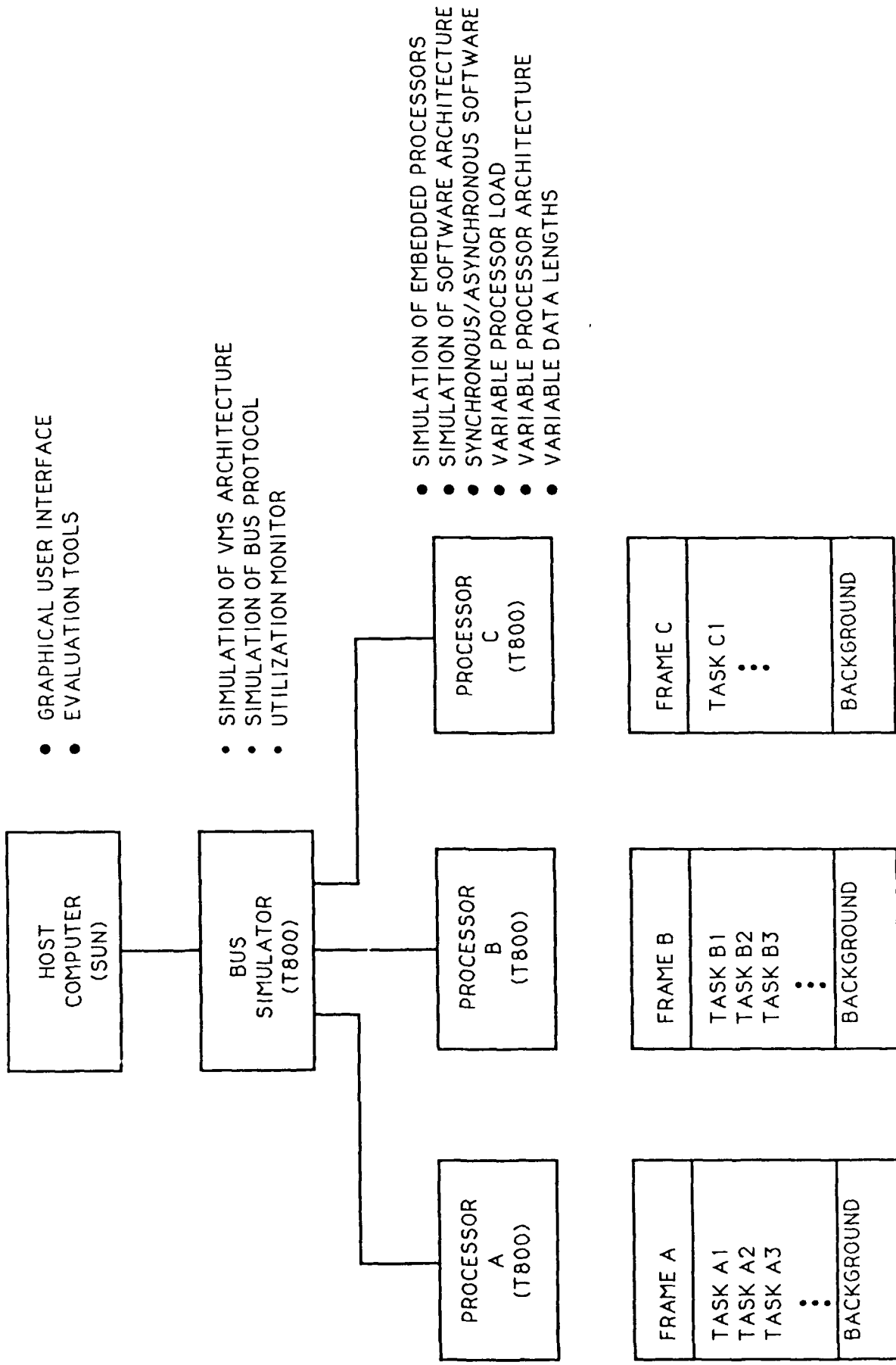


FIG. 2 PHASE I TRADE ARCHITECTURE



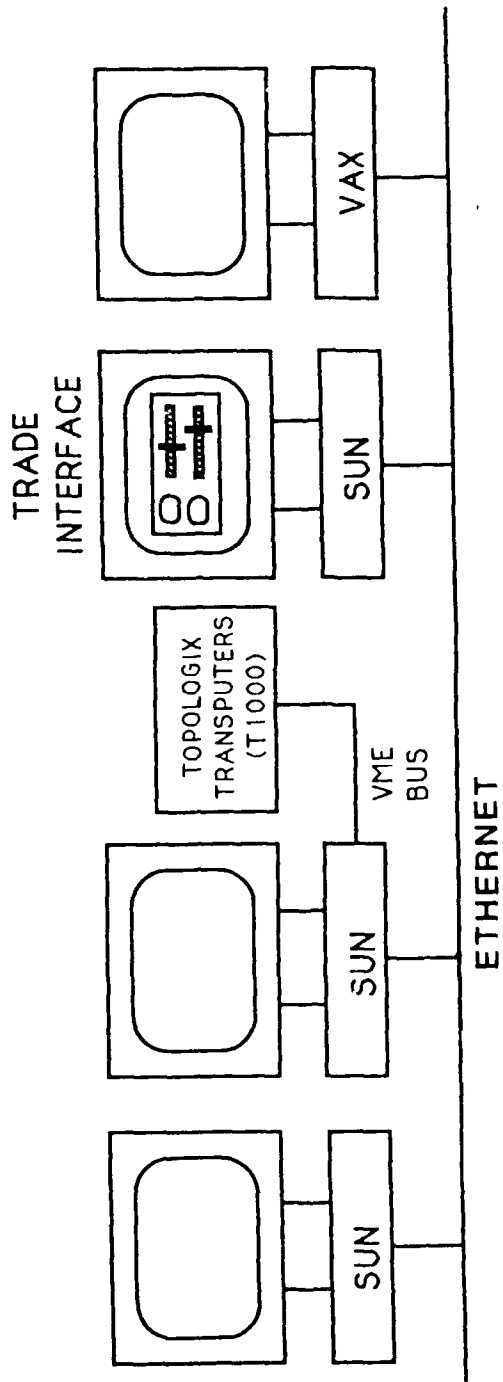


FIGURE 3. TRADE DISTRIBUTED WORKSTATION ENVIRONMENT

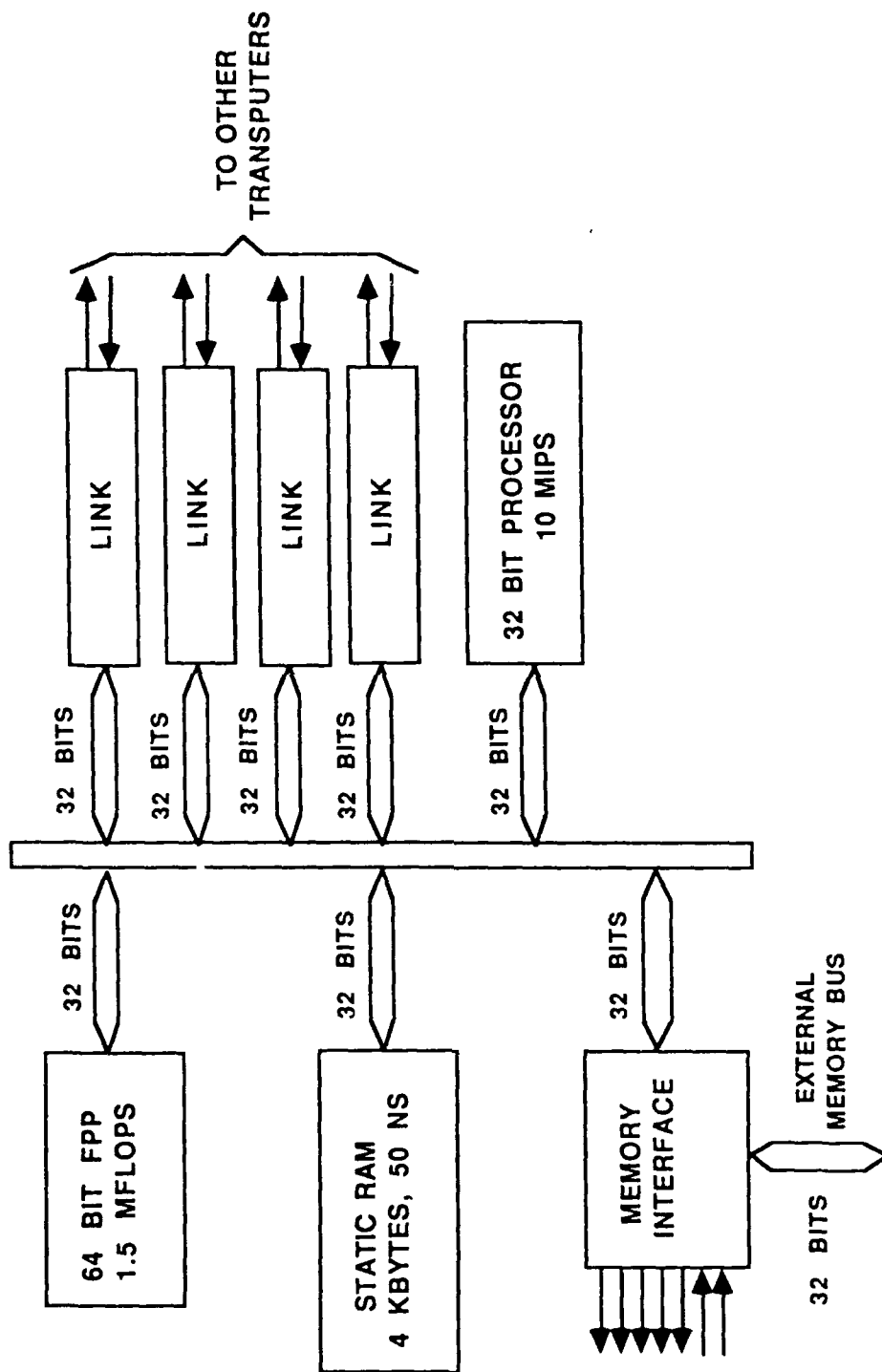
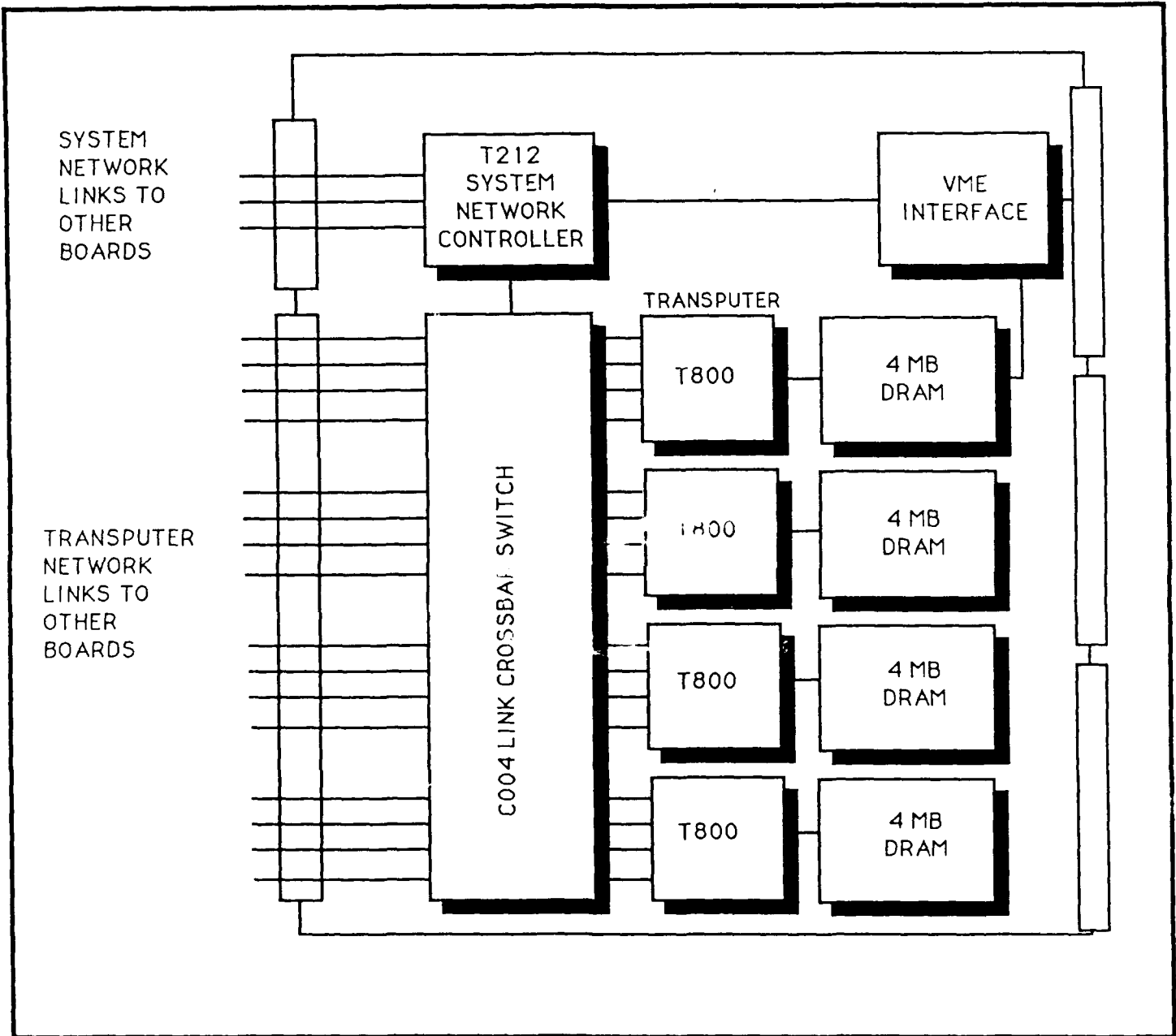


FIGURE 4. TRANSPUTER CHIP ARCHITECTURE



90/DM0705-06

FIGURE 5. TOPOLOGIX T100 MOTHER BOARD ARCHITECTURE

	Node 0	Node 1	Node 2	Node 3
* Bus				
0	-1	1	1	0
1	-1	0	1	1

FIGURE 6. BUS SPECIFICATION TABLE

```

*
*
*      PROCESSOR FORMAT FILE (PROCESSOR #1)
*
* Task   *Vars   Time(mms)   Send node   Send Bus   Receive
0        0       5,000       -1          0         -1
1        1      10,000       -1          0         -1
3        1      15,000       -1          0         -1
0        0       5,000       -1          0         -1
2        1      25,000       -1          0         -1
0        0       5,000       -1          0         -1
1        1      10,000       -1          0         -1
4        0      15,000       -1          0         -1
0        0       5,000       -1          0         -1
2        1      25,000       -1          0         -1
5        0      80,000        2          0          2
    
```

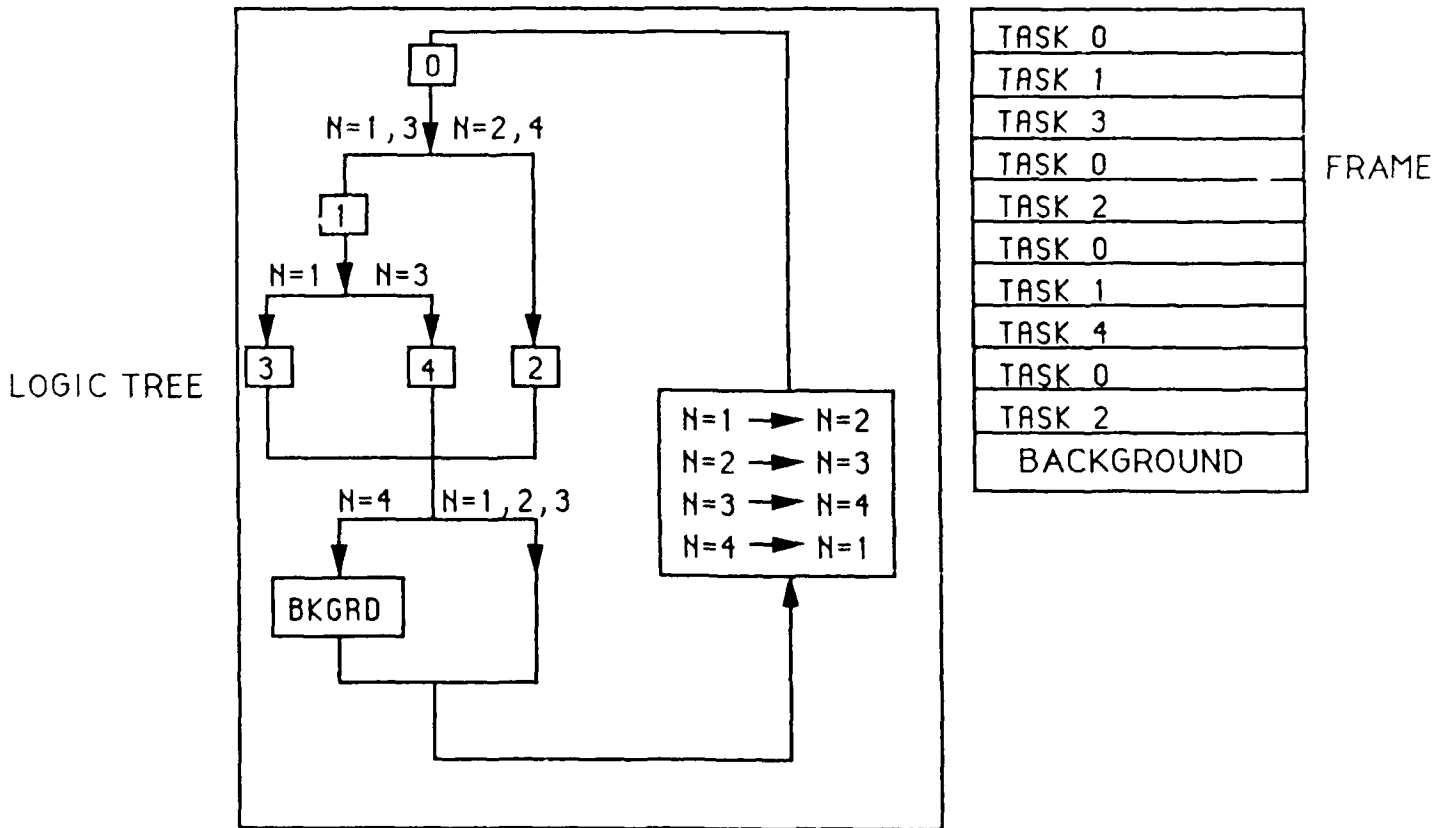


FIGURE 7. PROCESSOR FORMAT FILE

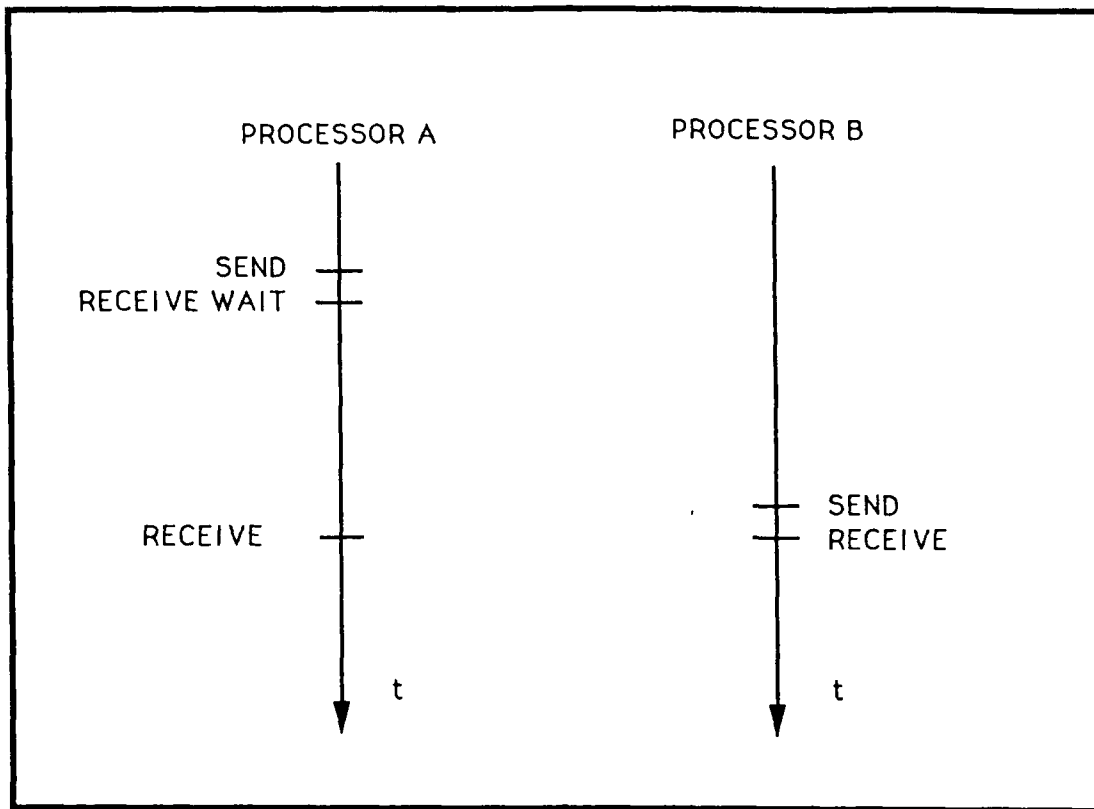


FIGURE 8. DUAL PROCESSOR SYNCHRONIZATION

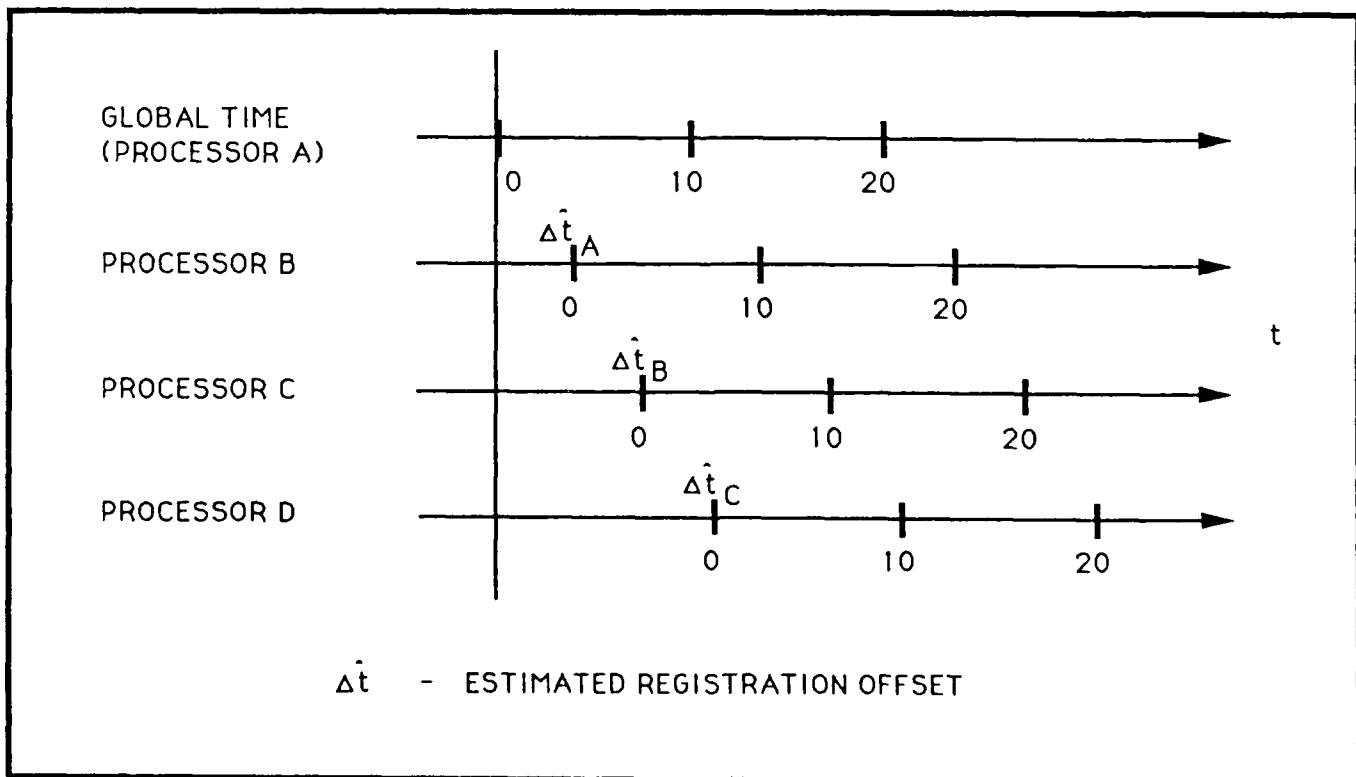
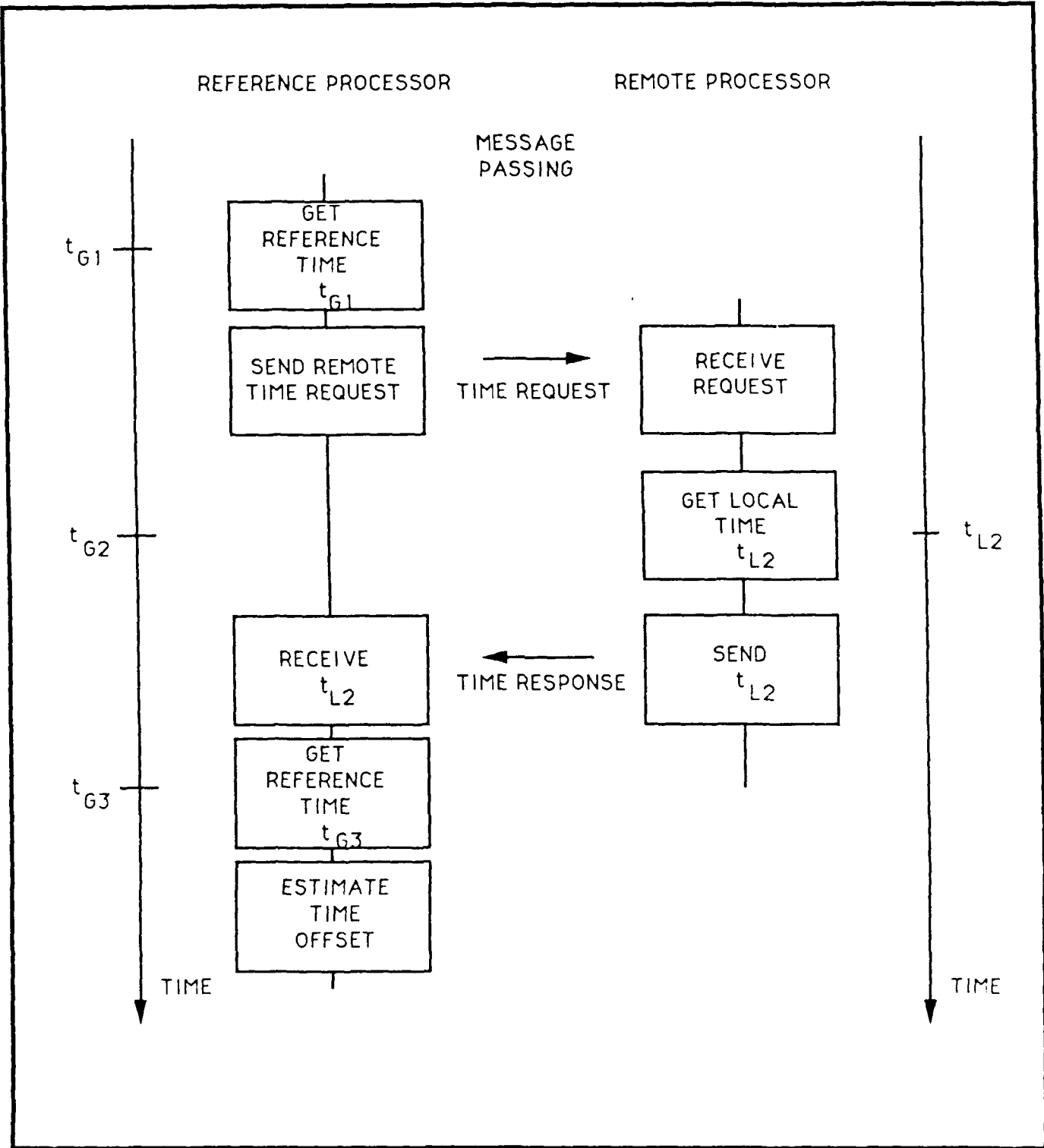


FIGURE 9. TIME REGISTRATION OF AUTONOMOUS PROCESSORS



90/DM0705-01

FIGURE 10. GLOBAL TIME REGISTRATION

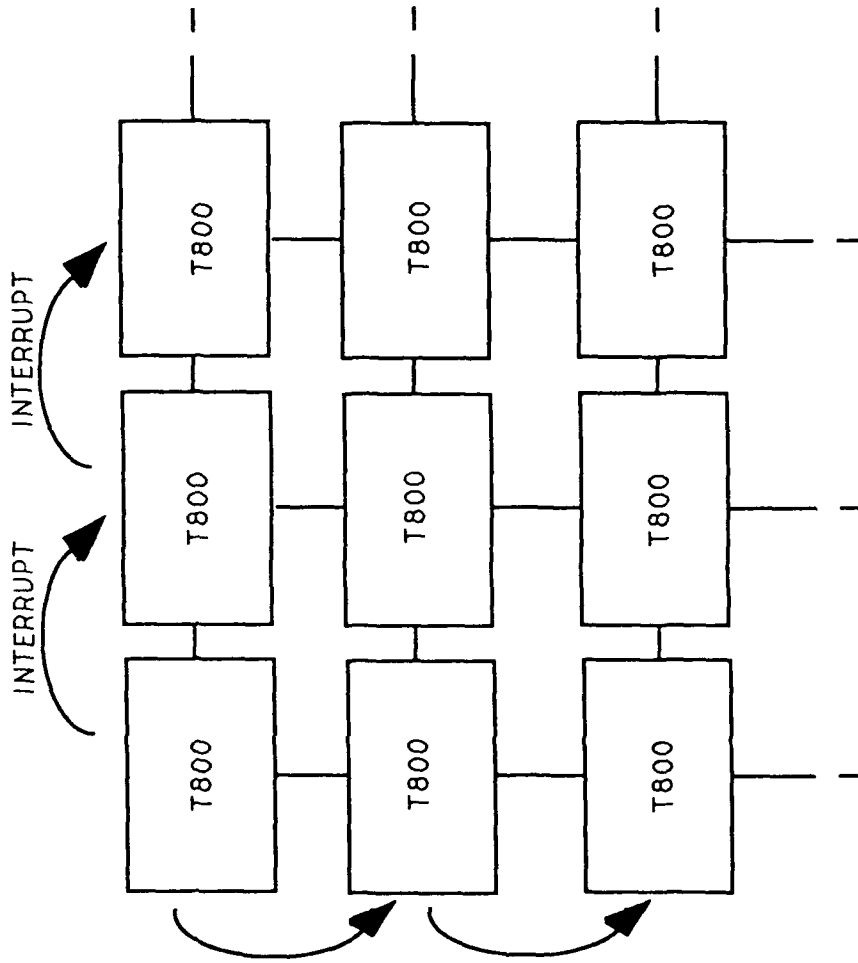


FIGURE 11. CASCADED INTERRUPT



VMS PARALLEL PROCESSING WORKSTATION, SPARTA INC.

**VMS Workstation Controls**

**Processor Topology**  
 Shared Bus  
 Pipe Line (2 Buses)

**Processor Architecture**

**Variable Tracking**  
 Node: 2

**Delay Histogram Settings**  
 Minimum Delay: 0  (ms)  
 Maximum Delay: 0  (ms)

**Overhead Settings**  
 Processor: 0  (mms)  
 Communication: 0  (mms)

Bus Baud Rate: 1000 (KHZ)

**Simulation Operation**

**Simulation State**  
 Bus Utilization  
 Bus A: 10.5 %  
 Bus B: 11.4 %  
 Process Overruns: 0  
 Number of Frames: 115

**Architecture Window**

**PIPE LINE BUS TOPOLOGY**

```

    graph LR
      BC[BUS CONTROLLER] --- N1[NODE 1]
      BC --- N2[NODE 2]
      BC --- N3[NODE 3]
  
```

**Delay Distribution**

FIGURE 12. TRADE USER INTERFACE