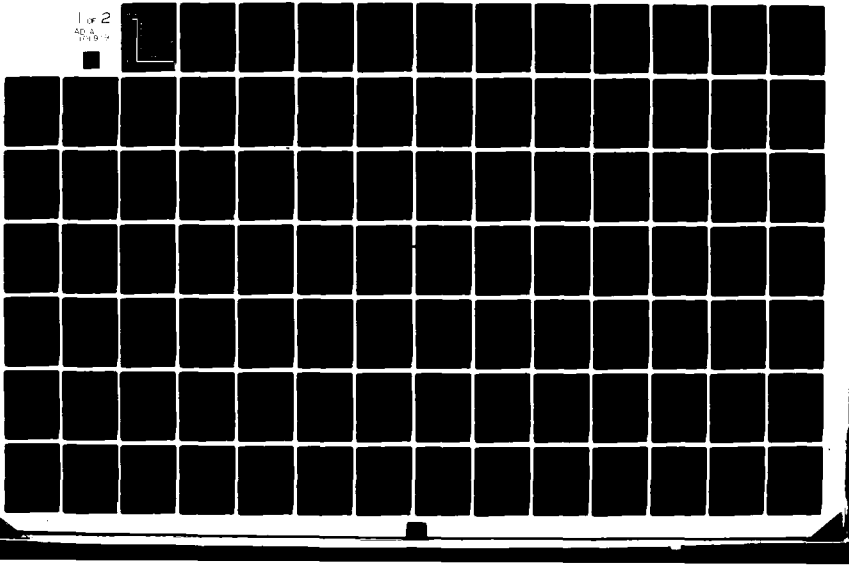
AD-A101 919 TELEDYNE BROWN ENGINEERING HUNTSVILLE AL SYSTEMS DIV F/G 9/2
ADVANCED MULTIPLE PROCESSOR CONFIGURATION STUDY.(U)
MAY 81 S J CLYMER F33615-79-C-0003
UNCLASSIFIED AFHRL-TR-80-43 NL



AFHRL-TR-80-43

AIR FORCE HUMAN RESOURCES LABORATORY

AD A101919

DTIC FILE COPY

ADVANCED MULTIPLE PROCESSOR CONFIGURATION STUDY

By

S. J. Clymer
Systems Division
Teledyne Brown Engineering
300 Sparkman Drive
Huntsville, Alabama 35807

OPERATIONS TRAINING DIVISION
Williams Air Force Base, Arizona 85224

May 1981

Final Report

AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235

NOTICE

When U.S. Government drawings, specifications, or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This final report was submitted by Systems Division, Teledyne Brown Engineering, 300 Sparkman Drive, Huntsville, Alabama 35807, under Contract F33615-79-C-0003, Project 6114, with the Operations Training Division, Air Force Human Resources Laboratory (AFSC), Williams Air Force Base, Arizona 85224. Patrick E. Price and Allen C. Snow were the Contract Monitors for the Laboratory.

This report has been reviewed by the Office of Public Affairs (PA) and 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.

MILTON E. WOOD, Technical Director
Operations Training Division

RONALD W. TERRY, Colonel, USAF
Commander

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFHRL-TR-80-13	2. GOVT ACCESSION NO. AD-A107 949	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* ADVANCED MULTIPLE PROCESSOR CONFIGURATION STUDY		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) S. J. Clymer		8. CONTRACT OR GRANT NUMBER(s) F33615-79-C-0003
9. PERFORMING ORGANIZATION NAME AND ADDRESS Systems Division, Teledyne Brown Engineering 300 Sparkman Drive Huntsville, Alabama 35807		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62205F 61142305
11. CONTROLLING OFFICE NAME AND ADDRESS HQ Air Force Human Resources Laboratory (AFSC) Brooks Air Force Base, Texas 78235		12. REPORT DATE May 1981 13. NUMBER OF PAGES 121
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)* Operations Training Division Air Force Human Resources Laboratory Williams Air Force Base, Arizona 85224		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)*

multiple processors
real-time computational evaluation
design performance predictor
flying training
simulation

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This document constitutes the Final Report for the Advanced Multiple Processor Configuration Study performed by Teledyne Brown Engineering (TBE) under Contract No. F33615-79-C-0003 with the United States Air Force Human Resources Laboratory (AFHRL). The overall objective of this study was to provide the Air Force with techniques for determining the effect of alternative multiple processor configurations on training simulator performance. This report summarizes the analysis and presents the baseline set of techniques identified by this study. Recommendations are made concerning the adaptation of these techniques for computation subsystem designs prior to their prototype development.

DD FORM 1 JAN 73 1473

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

PREFACE

Increased utilization of multiple processor configurations and the proliferation of mini- and micro-based processor capabilities/capacities permits a large number of alternative design configurations. Benefits of such designs include parallel processing, increased iteration rates, off-loading of specialized tasks, and integration with selected on-board avionics processing devices for flight simulation applications. This study was embarked upon to provide the Air Force with a generalized model of computer processor combinations from which flight training simulator computational candidate designs can be evaluated.

The work reported here was performed by the Systems Division of Teledyne Brown Engineering, Huntsville, Alabama. The work was done under Contract No. F33615-79-C-0003 for the Operations Training Division of the Air Force Human Resources Laboratory at Williams AFB, Arizona.

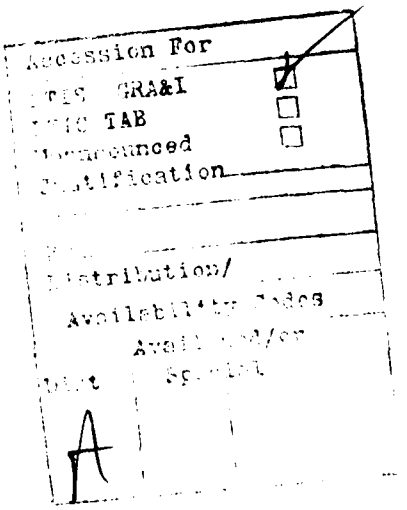


TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

List of Illustrations (*Contained*)

List of Illustrations (*Contained*)

LIST OF TABLES

ADVANCED MULTIPLE PROCESSOR CONFIGURATION STUDY

1. INTRODUCTION

1.1 Scope

This document constitutes the Final Report for the Advanced Multiple Processor Configuration Study performed by Teledyne Brown Engineering (TBE) under Contract F33615-79-C-0003 for the Air Force Human Resources Laboratory (AFHRL). This first section provides an introduction and an executive summary of the study in terms of objectives, background, approach and findings. Section 2 highlights the literature search and simulator analysis which led to the resultant design goals for the multiple processor performance prediction evaluation model presented in Section 3. Section 4 presents detailed design and implementation recommendations for automated tools with respect to evaluation of alternative candidate multiple processor configurations for flight training simulators. The concluding remarks of Section 5 summarize this study and list areas for further study.

1.2 Objectives

The major objective of this study was to provide the Air Force with techniques for determining the effect of alternative multiple processor configurations on training simulator performance. The techniques sought and addressed herein are applicable to both the optimal design process and the competitive design evaluation of simulator computational candidate configurations which may include mixtures of medium, mini-, and micro-computers and processors. For purposes of this contract, the term multiple processor configuration was defined to be any computational configuration containing more than one processor in which each processor is required to communicate (directly or via shared memory) with at least one other processor in the configuration in order to perform and coordinate its part of the real-time application.

1.3 Background

Increased utilization of multiple processor configurations and the proliferation of mini- and micro-based processor capabilities/capacities permits a large number of alternative design configurations. Benefits of such designs include parallel processing, increased iteration rates, off-loading of specialized tasks, and integration with selected on-board avionics processing devices for flight simulation applications. However, automated techniques for measuring performance and efficiency tradeoffs among many alternative design configuration were not available to the Air Force.

As a result, this study was embarked upon to provide the Air Force with a generalized model of computer processor combinations from which flight training simulator computational candidate designs can be evaluated.

1.4 Approach

An analysis of real-time flight simulation was performed. This analysis produced a set of design characteristics in terms of (a) flight simulator configurations and (b) multiple processor configuration performance measures. To support this analysis, a literature search was conducted and documented to provide state-of-the-art assessment of multiple processor performance evaluation tools and techniques.

The results of this analysis are documented in Section 2 of this report. These results include recommended configuration considerations for flight simulator computational environments.

The analysis laid the foundation for a set of multiple processor design evaluation goals. These design goals were then expanded in terms of outputs, inputs, and model processes necessary to facilitate evaluation of various configurations of multiple processors. The design goals and model feasibility are described in Section 3 of this report. The model design is described in Section 4.

1.5 Results

TBE has identified a baseline set of techniques and tools which can provide for a systematic evaluation of alternative candidate multiple processor designs with respect to a set of quantitative computational measures for a given application. Implementation of these techniques for simulator design evaluation is enumerated in Section 4 of this report. As with any tool or technique, the skill of the personnel using them and the availability of data are key issues in selecting and implementing them as part of a standard evaluation procedure.

It is also important to note that computational design performance is just one area of flight training candidate design evaluation. The evaluation of new algorithms and special-purpose equipment to meet training configuration requirements generally requires some level of prototype implementation to properly assess human factors.

2. FLIGHT SIMULATOR ANALYSIS

2.1 Literature Search

In order to provide a relevant set of flight simulator candidate design evaluation techniques, TBE performed a literature search to collect data on current trainer computational configurations and generic multiple processor real-time performance features. Figure 1 illustrates the key word form utilized to search for and characterize the collected articles for detailed analysis.

Specific flight simulator configuration materials were collected from both military and commercial organizations. Initial compilation of a list of flight simulator facilities (tabulated in Table 1) was derived from analysis of several key simulator facility survey articles (References 1, 2, 3). These facilities (Table 1) were then contacted (via phone) to obtain up-to-date configurations data with respect to the following items:

1. Aircraft modeled
2. Training/engineering positions
3. Computational configuration to include processors, memories, mass storage, and development language
4. Special-purpose subsystem peripherals to include visual imagery, motion base, "g" queuing, instructor station, and training station equipment features.

Most of the military-related organizations contacted sent facility capability brochures. These brochures provided high-level descriptions of the computational configuration.

PARTITIONING STUDY
REFERENCES

TITLE: _____

_____

AUTHOR: _____ DATE: _____

ORIGINATING AGENCY: _____

CONTRACT NUMBER: _____ DOCUMENT NUMBER: _____

TBE FILE NUMBER: _____ CLASSIFICATION: _____

LEVEL 1 INDEX:

_____ FLIGHT TRAINER SYSTEMS
_____ REAL TIME PROCESSING
_____ COCKPIT CONTROLS/SWITCHES/ INSTRUMENTATION
_____ MOTION
_____ VISUAL
_____ AUDITORY
_____ FORCE
_____ COMPUTATIONAL
_____ TERRAIN MAP
_____ TRAINING
_____ ON BOARD COMPUTERS
_____ NAV/COMM
_____ MALFUNCTIONS
_____ FLIGHT
_____ ENGINE
_____ ON BOARD WEAPON SYSTEMS
_____ FUNCTIONAL FLOW
_____ HQ CONFIGURATION
_____ SW OPERATING SYSTEM

OTHER _____

LEVEL 2 INDEX:

_____ SW APPLICATIONS
- _____ SW TASK DESCRIPTION
- _____ SW TASK FLOW
- _____ SW TASK CODE
- _____ SW TASK ITERATION RATES
- _____ SW TASK DEPENDENCIES
- _____ DATA BASE PARAMETERS
- _____ DATA TRANSFER
- _____ DATA STORAGE

_____ HW COMPONENTS
- _____ PROCESSOR
- _____ MEMORY
- _____ DATA BUS
- _____ COCKPIT
- _____ VIDEO DISPLAY
- _____ CRT DISPLAY
- _____ MOTION BASE
- _____ AUTIO EQUIPMENT
- _____ DISC, TAPE
- _____ A/D & D/A
- _____ KEYBOARD, TELETYPE
- _____ PRINTER
- _____ RECORDING EQUIPMENTE
- _____ CARD READER
- _____ g – EQUIPMENT

Figure 1. **Keyword Literature Search Form used to categorize literature collected for simulator analysis.**

Table 1. **List of Facilities Contacted**

Facility	Point of Contact	Facility	Point of Contact
ASPT	AFHRL Flying Training Division (AFSC) Williams Air Force Base, AZ	FSAA	NASA/AMES Moffett Field, CA
A7-E NCLT	T.D.C. Cecil Field, NAS, FL	F-15A	Goodyear Aerospace Corporation Akron, OH
Boeing 707	Boeing Commercial Airplane Company Seattle, WA	Lambs	Vought Aeronautics Co. Dallas, TX
Boeing 727	American Airlines Ft. Worth, TX	LAS/WAVS	Northrop Corporation Hawthorne, CA
727	Continental Airlines Los Angeles, CA	MACS, I, II, III	McDonnell Douglas Corporation St. Louis, MO
Boeing 727—200	Braniff International Dallas, TX	MACS-Device 2E6	McDonnell Douglas St. Louis, MO
Boeing 737	United Airlines Denver, CO	P-3C	NAS/Moffett Field, CA
		S-3A WST	T.D.C NAS/Cecil Field, FA
DC-8 Instrument Flight Sim.	Flying Tiger Airlines Los Angeles, CA		
		S-61 VMS	NASA/LANGLEY Research Center, VA
DC-8	Braniff International Dallas, TX	SAAC	Luke AFB, AZ
DC-10	American Airlines Ft. Worth, TX	SH-2F HWST	NAS/Norfolk, VA
DC-10	Flight Safety International Long Beach, CA	TA-4J	NAS/Chase Field Beeville, TX
Differential Maneuvering Sim.	NASA/Langley Research Center, VA	T-2C	Same as above
F-14	TD 1 HAS/Miramar, CA	VACS	Vought Corporation Grand Prairie, TX

In addition to the specific flight simulator configuration materials, we collected general materials regarding selection, design, and performance evaluation of multiple processor configurations for real-time application processing needs versus utilization of processing features unique to given candidate hardware configurations. The majority of articles which relate to performance prediction topics tend to come from professional technical journals and proceedings such as AIAA, ACM, IEEE, and others.

The resulting analysis of current flight simulator configurations is presented in Section 2.2 and 2.3. This is followed by the generalized analysis of real-time application multiple processor performance features, design goals, and performance evaluation modeling techniques which are presented in Section 3.

2.2 Computational Interfaces

Flight simulator facilities may be categorized into three distinct operational environments; namely, airframe research/development, training research/development, and training simulator devices. This contract was primarily concerned with the hardware configuration features which are required to satisfy and support the computational aspects of the real-time operational flight training simulator device environment.

In order to fully appreciate the computational hardware configurations, one must first consider the major external interfaces which are serviced. At the highest level, the flight training device configuration must coordinate and respond to "man-in-the-loop" directives and responses. Figure 2 depicts the generic manual interfaces as trainees, instructor/operators, system development analysts, and operational maintenance analysts. The manual procedures are not within the scope of this contract; thus, emphasis is placed upon the hardware/software systems discrete interfaces with man which can be parametrically and quantitatively defined.

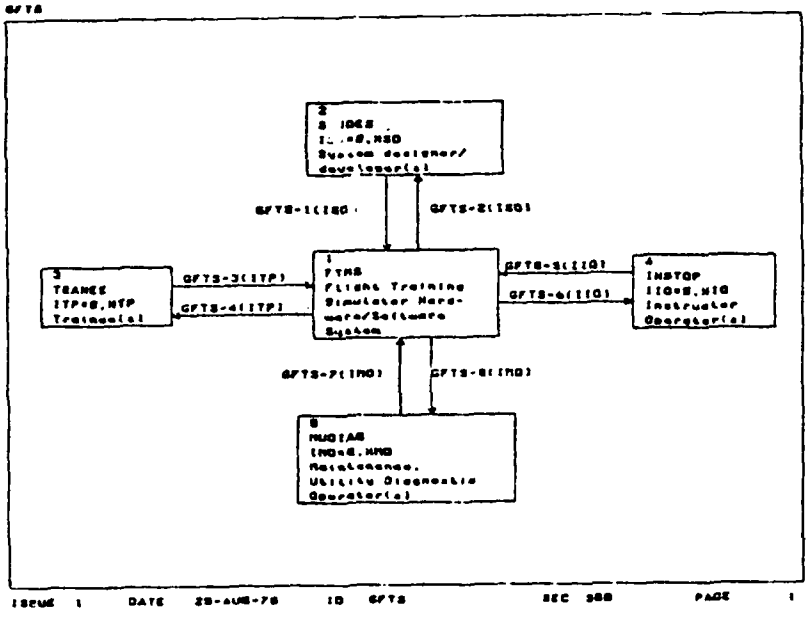


Figure 2. **Generic "man-in-the-loop" interfaces with a flight training simulator.**

It should be noted that the categories of manual interfaces have been selected with respect to major functional interfaces. An individual may interact with the system in one or more roles. For example, a system developer can submit a test execution of a new or modified capability which requires inputs from instructor/operator stations and/or training position devices. To expedite initial checkout in the test environment, system developers may "act" as instructor(s) and/or trainee(s) to test these interfaces. A trainee in the self-instruction mode will utilize instructor/operator controls (as well as the training station equipment) to select training exercises and obtain performance evaluation reports. An instructor may "act" in a trainee role to test, record, and evaluate the simulation system responses to the training position

controls. In summation, the system must accommodate and supply the basic tools and interfaces by which trainees, instructor/operators, maintenance operators, and developers can interact. To place the current configuration analysis in proper perspective, each of these "man-in-the-loop" interfaces is briefly expanded.

2.2.1 *Trainee Positions.* The trainee positions supported by a training simulator configuration (TRANEE of Figure 2) may be as simple as one pilot to the complexity of flight crews for multiple aircraft systems. For purposes of this analysis, training positions have been categorized to include pilots, co-pilots, on-board computer operators, on-board weapon system operators, or any combinational positions which require manual interaction in the real-world system being simulated. Each distinct trainee position has a finite number of controls, dials, and switches which provide real-time inputs to the computational configuration. These inputs are monitored at prescribed frequencies by the computational subsystem which, in turn, must output a realistic simulated environment to include instrument readings, on-board displays, motion, gravitational forces, visual, audio, navigation, and communication responses.

2.2.2 *Instructor/Operators.* An instructor/operator (INSTOP of Figure 2) interface with the existing simulation facility is to conduct, monitor, and evaluate training. In the generic system, independent simultaneous training sessions may be taking place. Each one may use one or more instructors and/or operators. These operational input interfaces have been categorized in the following groups:

1. Initialization for a specific training configuration
2. Change mode of training job (real-time state, scenario record/playback, etc.)
3. Display/update training data base parameters
4. Insert malfunctions
5. Insert weather and environment data
6. Request monitor displays
7. Display aircraft indicators on instructor panel
8. Request performance evaluation report
9. Voice transmit/receive/record.

The types of outputs may be generically categorized as:

1. CRT/video/plotter displays
2. Printer/teletype summaries and listings
3. Instructor panel indicators, lights, and warnings
4. Audio communications.

As with the training position station, an instructor station will consist of a finite set of entry devices. The flight training system is required to monitor each control/switch/keyed entry at prescribed frequencies. When more than one instructor is participating in a training session, priorities and override interfaces must also be defined. Resource allocation requirements are necessary when simultaneous independent training sessions are being conducted and monitored by the same instructor station.

2.2.3 *Maintenance Operators.* Day-to-day operations include routine services as well as troubleshooting diagnostics and repairs when needed. Routine services include powering the system up/down, mounting tapes/discs/paper, scheduling job entry and special duties associated with equipment readiness and preventive maintenance. When system problems arise, system troubleshooting procedures for HW/SW fault isolation are performed. Maintenance HW/SW features must be an integral part of system performance and utility evaluation with respect to anticipated down time, maintenance costs, and support software/hardware resources.

2.2.4 *System Developers.* The training simulator configuration must be amenable to inevitable changes in aircraft configuration and/or procedural changes introduced during its operational life cycle.

To do this, system development/maintenance personnel must be provided appropriate interfaces for introducing, testing, and incorporating new/changed modules/subsystems in areas of software and hardware. These interfaces with the computational subsystem must be performed in accordance with established configuration management/quality assurance controls to assure maintenance of an operationally ready training facility.

2.3 Computational Configuration Features

To provide responsive real-time interaction with the interfaces described in Section 2.2, the simulator configurations studied employ multiple processors to handle the real-time process load. In general, no single processor currently exists with the capacity to handle all advanced training simulation computational needs. The numerous alternatives for hardware selection, interfacing devices, and process/storage partitioning provide a complex maze of configuration design evaluation decisions.

Certain common computational design features were extracted during the analysis of flight simulator designs. Those configuration features identified include:

1. Operational Real-Time Command, Control, and Communication (C^3)
2. Diversified I/O
3. Math Model Arithmetic Precision, Accuracy, and Stability
4. Sufficient System Spare Capacity and Growth Provisions
5. Reliable and Proven Hardware
6. Development and Diagnostic Support

Each of these factures is now expanded in terms of:

1. Related real-time computational environment needs
2. Applicable statements extracted from USAF Military Specification (MIL-D-83468) Digital Computational System for Real-Time Training Simulators
3. Observed implementation characteristics in current flight simulator configurations.

The relationships of these features to performance prediction measurement is given in Section 3.

2.3.1 *Operational Real-Time Command, Control, and Communications (C^3).* The success or failure of a given multiple processor configuration relates to the ease with which it adapts to a variety of flexible training configurations. Figure 3 illustrates major functional areas which must be coordinated by a real-time flight training simulator executive program. This real-time customed executive program coordinates the functional module execution rates and data transfers for a given training job. With the exception of the initialization interfaces, these functions require high iteration rates with complex real-time data base update coordination to produce a conducive training environment for all participants.

In general, the more training and instructor station positions supported, the more complex the enumeration of potential training configuration and required C^3 synchronization becomes. A multiple processor network configuration for flight training simulators must have real-time operational features which support control of the current training mix via straightforward user interface commands which are properly communicated in terms of resource sharing, task iteration and sequencing, and peripheral device I/O servicing. When multiple independent training sessions are permitted, a means for system resource sharing, as well as a given training job resource sharing, is necessary in terms of multi-tasking, data base retrieval, processor assignments, memory management, processor-to-processor communications, and peripheral device assignments.

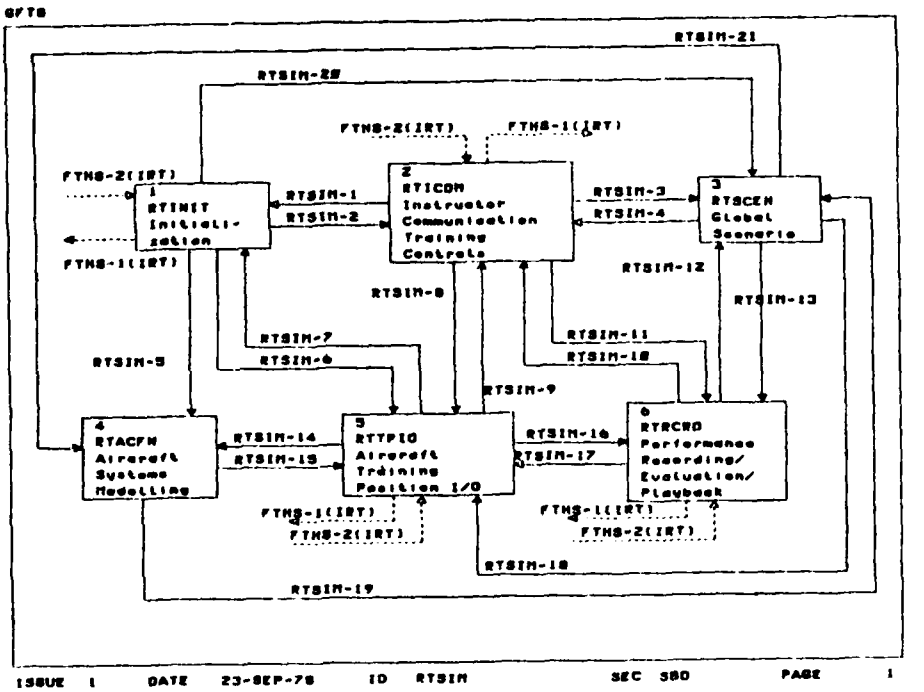


Figure 3. **Real-time executive must coordinate all task and data flow for given functional areas.**

MIL-D-83468 (USAF) states the following regarding operational control:

> "3.1.1.1 *Multiprocessor/multicomputer configuration.* If a multiprocessor/multicomputer configuration is proposed to meet the operational requirements of the trainer, the following requirements shall apply to each computer in the system. Multiprocessor/multicomputer configurations shall be designed such that all computers operate in parallel in real-time and are controlled and time synchronized from a single computer program supervisor/executive. This supervisor/executive shall direct the problem flow and establish priority controls."

In the past, many of the control functions have traditionally been handled via a custom-tailored, real-time flight training simulator executive driven by hardcoded software structures which limited themselves to a predefined set of training configurations and resulted in hard-to-reconfigure designs. The current design trends are toward more modular executive structures which are device and processor independent. Still, the real-time constraints are very demanding, requiring a means of streamlining executive communication and controls. As a result, the basic vendor-supplied real-time operating systems have been expanded to incorporate many of the previously defined executive functions, permitting the actual executive to be constructed as a series of macros which contain C^3 parameters in generic form to support a more flexible system executive environment. This flexibility does reduce the ratio of application code to system supplied code.

2.3.2 *Diversified I/O.* A flight training simulator incorporates a variety of training station I/O devices to include aircraft controls, switches, instrumentation, head-up display; weapon system controls, switches, displays; on-board computers, keyboards, displays; plus simulated out-the-window scenes, motion, "g" forces, sound (aircraft, engine, weapons, etc.), and radio navigation and voice communication. The proper coordination of these components is a function of the training maneuver(s) bing performed (air-to-air combat, air-to-surface combat, going to and coming from mission, takeoff and landings) and the conditions being simulated/recorded (task difficulty, weather conditions, malfunctions, and instructor station monitoring and scoring features). These are in addition to standard computer configuration components consisting of tapes, discs, operator stations, processors, memories, card readers, printers, and CRT terminals. The flight training simulator candidate configuration must satisfy the diverse I/O requirements including A/D sampling input rates, D/A response output rates, and digital communications with a large number of unique devices.

To permit maximum training configuration flexibility, MIL-D-83468 states that for multiprocessor/ multicomputer configurations:

> "Each computer shall be capable of directly communicating, without involving other central processing units (CPU's), with all peripheral equipment, all instructor station systems for which it computes information or controls information flow, and all interfaces for which it computes or controls information."

Specific input/output capabilities are further expanded in paragraphs under 3.1.2 of MIL-D-83468.

The means for accommodating special-purpose device I/O is directly related to the application software partitioning problem. The design studied included two major approaches, namely:

1. Partition of processors into functions such as visual, math model, instructional aides for all training stations with provision for high-speed transfer of interfacing data blocked by postion hierarchy structures.
2. Partition of processors into self-contained groups supporting all functions for a given training/ instructor station and provisions for high-speed data transfers when interactive (one-on-one, two-on-one, etc.) training positions must be correlated.

Of the configurations studied, the interactive capabilities were limited to one-on-one and two-on-one combat/formation maneuvers. Physical processing timing/sizing problems are apt to occur for either of the above partitioning schemes when more than three interactive aircrews are being trained. Thus, a communication matrix network is needed to provide both functional and/or positional related data. The partitioning tradeoffs are numerous in themselves and were the subject of a related report (Reference 7). For purposes of evaluating multiple processor performance, the software partition is assumed to be well-defined in terms of the candidate configuration. Thus, the major concern becomes one of ensuring that all the special-purpose I/O devices are being serviced in a timely and proper fashion. Another I/O issue concerns degraded mode alternatives for reconfiguring the processor loads and/or memory storage interfaces to service given peripherals when a processor and/or memory unit must be taken off-line. Further study of this area is recommended.

2.3.3 *Math Model Arithmetic Precision, Accuracy, and Stability.* The numerical precision (number of significant digits) and accuracy (units) is a vital part of the simulated aircraft models which incorporate preset flight data and device biases for real-time response calculations based on trainee control inputs. Early versions of digitally driven simulators employed fixed binary arithmetic making it the coder's responsibility to keep track of scaling and accuracy for all arithmetic operations. As floating point hardware proved itself to be adequate with respect to both timing and accuracy, the mathematical models

have been redesigned for use with floating point. Reference 4 is a dissertation which specifically addresses the cost-effectiveness of floating point hardware operations with respect to the real-time simulation problem. Since this publication, significant improvements have been made which make floating point hardware even more advantageous along with lower hardware costs. One of the major cost-effective reasons pertains to the decreased number of instructions which must be coded/debugged/maintained for a given real-time simulation.

MIL-D-83468 outlines areas of numerical stability and accuracy which must be analyzed with respect to the dynamics of the problem plus specific processor truncation, roundoff, and propagated errors. These areas include numeric integration techniques, intervals, and interation frequency of associated I/O parameters utilized by the various integration models. This specification also requires any floating point operations be performed via floating point hardware as opposed to softcoded equivalents.

Current utilization of a fixed word length machine with byte, half-word, full-word, and double-word data structure characterization permits a variety of data operations in both fixed and floating point instruction sets. The precision and accuracy features for incorporating these flexible structures must be addressed as a design issue in terms of what particular precision (number of significant digits) is required to satisfy computational relationships for the desired accuracy (units). In some instances, the automatic assumption that floating point is adequate may lead to unstable numeric conditions if some scaling analysis has not been performed. This is due to the gross discontinuities when operating on additive floating point parameters which have a substantial range in order to magnitude. In addition, the trend in higher-order language specifications is one of specifying the required precision and accuracy as opposed to using default word size of a given target machine. Significant increased speeds of future hardware logic may make floating point binary coded decimal with automatic machine scaling arithmetic logic units a future consideration. In essence, the task data base design must evolve from a numerical analysis of the parameters and the intermediate relationships of the calculations necessary to obtain system responses which are maintained in a certain precision to guarantee a minimum level of accuracy. Thus, the selection of processors and memories becomes one of matching arithmetic features with required design arithmetic features for each of the various functional tasks.

Current simulator designs show a trend toward use of 32-bit floating point operands as opposed to earlier (1960's) designs which were 16-bit fixed point. They also employ a varied range of iteration rates. This has resulted in a combination of increased hardware real-time capabilities and extensive operational R&D test bed "fly-offs" among various math model computational algorithms.

2.3.4 *Sufficient System Spare Capacity and Growth Provisions.* The real-time computational tasks have a mixture of memory, processor/peripheral iterface needs. Memory storage is required in the following categories:

1. Preset tables of flight data for real-time parameter interpolation for each aircraft or weapon system being modeled
2. Training device bias value tables to compensate for system calibration idiosyncrasies
3. Shared block of intermediate simulated state data
4. Instruction storage
5. Temporary computational storage.

The real-time processing cycle is generally characterized as a set of tasks with prescribed iteration rates, sequencing, and data dependencies. Thus certain tasks are permitted to execute in parallel whereas others must be serially sequenced in the allotted real-time cycle. As more training capabilities are added, more system resource are needed. This is especially true when additional trainin device stations are added to the configuration. The candidate configuration memory parameters must properly relate private and/or shared memory capabilities with respect to processor and peripheral device communication buffers.

Shared memory is generally organized around a multiport controller which allows a specific port assignment for given CPU access to the common data base and system status information.

A critical part of any candidate design configuration is the provision for spare capacity within each individual processor, memory mass storage, or I/O interface unit. MIL-D-83468 paragraph 3.1.6 addresses specific measures, spare processor, memory, on-line mass storage, and I/O channel interface capacity. Spare growth capability measures are described in paragraph 3.1.7 of this MIL SPEC. Currently 100% expansion capability is required in the areas of directly addressable memory, mass storage, processing capacity, and input/output. In addition, the growth capability must be achievable with minimal design changes to existing hardware/software and equipment.

To account for spare capacity and configuration growth, the computational real-time cycle tasks are allocated via a storage and timing budget. These budgets should incorporate the task iteration rate, data dependencies, processing dependencies, and communication interfaces. However, in many of the observed design documents these numbers were represented as aggregate totals including system overhead rather than at the task level. In most cases, these budget grow larger as design and implementation details become known. This is another major reason for building in large spare and growth factors in the conceptual and preliminary design phases. It also supports the theory of refining the software design prior to a final commitment to specific computational equipment and internal configuration interface selection. One Air Force evaluator expressed a major need in this area for better tools and techniques to help determine if proposed configurations are adequate and, also, to evaluate major change proposals to add processors and/or memories when the revised budgets indicate spare capacity or growth violations.

Another feature which relates to spare/growth configuration design concerns task partitioning problem. For example, if multiple duplicate aircraft models are involved in the training sortie, then certain common blocks of aircraft data are needed to support computations for the duplicate training positions. In addition, a training position current state block will be needed for each of the duplicate training positions. These current state blocks are most likely to be in private momories associated with the specific processor performing the aircraft model computations for the training device whereas the basic aircraft model data will most likely reside in a shared memory. Thus a tradeoff between hardware/ software design solutions must be made.

2.3.5 *Reliable and Proven Hardware.* Operational flight training requires coordination of facility, instructor, operator, maintenance, and student personnel schedules. Postponement of a training exercise due to equipment failure can be a costly rescheduling problem, particularly in a temporary duty training environment where personnel must extend their TDY assignment or reschedule a TDY assignement for a later date, thus impacting other duty assignments and possibly slipping project milestone accomplishments.

Specific availability, maintainability, and reliability requirements are specified by the Air Force for a given simulator procurement in terms of the total simulator system. For example, the simulator facility may be required to be available for training 16 hours per day, 6 days a week for a 20-year life cycle.

A viable multiple processor candidate design must be composed of proven, reliable components. Hardware technology has made significant advancements with respect to component preventive maintenance, meantime between failure, and parts replacement schedules. As new technologies become available, the feature of reliable and proven hardware must be analyzed for a given design implementation based upon laboratory prototype test conditions of the technology components. This is necessary to reduce the amount of risk associated with the introduction of new technologies into the operational environment.

In addition to individual component reliability, the total system must be viewed from a master schedule of expected operational versus maintenance timelines which considers the net effect of the various component schedules. This generally entails the combining of prventive maintenance, parts replacement schedules, and spare supplies to consolidate the anticipated system downtime. A candidate design must include the anticipated schedules along with the supporting analysis inputs which were used to derive the schedule. One problem in this area is assuring availability of spares and decisions as to when to upgrade equipment per periodic vendor modifications/enhancements.

The govenment typically assumes the operational system maintenance upon acceptance of a training simulator device. As a result, many logistic problems from past experience are now required to be addressed as part of the design specification. Use of built-in test equipment (BITE) and built-in-test (BIT) procedures are becoming an essential design feature for efficient maintenance of advanced computational subsystems. Reference 5, "Downtime Wastes the Resources," provides an overview of the logistics support problems and their impact on life-cycle costs (LCC).

2.3.6 *Development and Diagnostic Support.* Development of a new training simulator device represents a major investment in both time and money. An essential requirment for a training simulator device is that it be representative of the operational device(s) for which training is being conducted. Operational configurations are continuously being upgraded to meet new mission requirements and/or enhance the manual operation control response features. Therefore, the training simulator device must be flexible and modular in design to permit software/hardware upgrades in a timely/orderly manner.

Before any new application/change can be realized, it must be conceived, specified, designed, built, and successfully tested. A number of software tools can provide a systematic means for carrying out the system development cycle. Once an operational capability has been developed, a means for maintaining it is required. Automated maintenance software can facilitate systematic procedures for configuring the system or a subsystem required to support specific operational real-time training jobs. Figure 4 represents the Flight Trainer Hardware Software (FTHS) in terms of the above defined jobs which must be coordinated and allocated via a system resource manager to include:

1. Real-time training simulation jobs
2. Development jobs
3. Test jobs
4. Maintenance jobs.

At this time, it should be pointed out that no link to specific hardware configuration has been made. The development tools could reside on a separate computer system which formats files for the object system where actual training occurs; or, the tools could reside as an integral part of a multiprocessor system for development, upgrading, and training.

Section 3.2 of MIL-D-83468 addresses the computer program system components and features which are required to facilitate total simulator operation, maintenance, training, and upgrade configuration controls. Emphasis is placed on a top-down design incorporating modules which can be identified and handled in an independent manner. Standard vendor-supplied support programs are considered to be deliverables including the resident operating system, peripheral operating/handler systems, loaders, assemblers, compilers, memory dump programs, mathematical library, copy routines, and trace routines. In addition, a full set of maintenance and test programs are to be supplied including computer diagnostics, real-time interface equipment diagnostics, discrete I/O tests, analog I/O tests, program control test of real-time clock, simulator equipment check, and calibration test programs. Where possible, standard vendor-supplied programs are to be utilized for these functions and any changes must be justified.

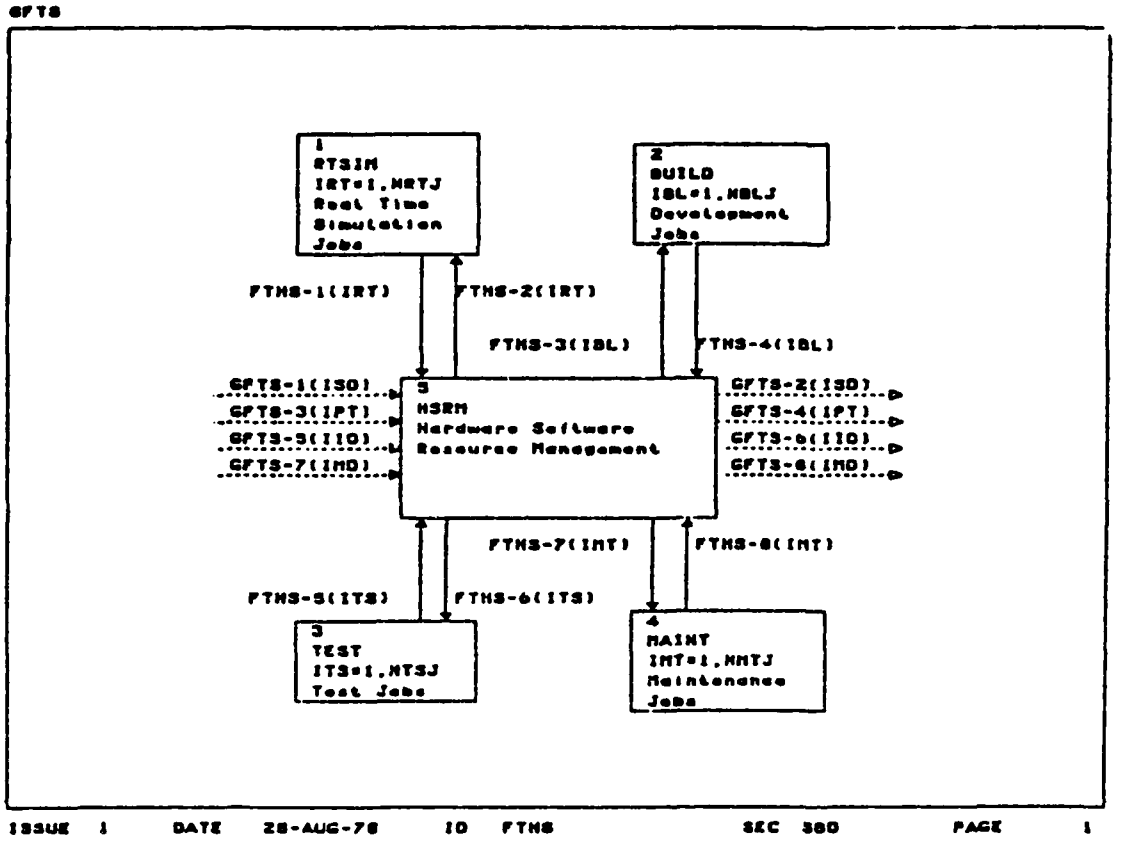


Figure 4. **Generic Job Categories include the real-time simulator plus support aids for development, testing, and maintenance of system upgrades and modifications.**

In the software development of recent simulators, such features as real-time operating systems, virtual memory management, optimized high-order language compilers, instruction set hardware options, and data base management utilities have reduced the amount of application dependent code which must be written. Another important feature is the array of tools available for configuring and maintaining software source to obtain resulting operational code. These tools instrument source editing, library creation, and multiple baseline code releases. Other software tools facilitate documentation via cross-reference listings, flowcharts, indexing, and flagging code which is not documented or does not adhere to the coding standards. If security-sensitive models are incorporated in the system (i.e., weapon system performance data), other features which can not be overlooked are user and group file access and update protection via authorized accounting and password system procedures including external on-line terminals.

In ongoing developments, there is an increased trend to micro-code frequently used software functions, such as linear function interpolation (LFI). These selected so-called "firmware" features are then considered to be "hardwired" in a read only memory (ROM). There is some professional disagreement as to just how firmware should be designed, documented, and maintained. The proliferation of erasable and programmable ROMs and writable control stores (WCS) devices is a testament to the fact

that firmware designs do not remain static once defined and implemented. Their use does permit efficient solutions to time-critical function execution. However, it is important to have detailed design documentation concerning the flexibility and/or limitations incorporated from an application programmer's viewpoint. If vendor-supplied ROMs are incorporated and changes are anticipated, the nature of the changes and the party responsible for making and maintaining configuration changes should be clearly delineated.

Another software support design issue concerns the best way to incorporate and/or simulate on-board processing device functions and subsequently how to introduce and maintain upgrades associated with these devices. The discrete training modes of freeze, playback, advance, plus evaluation scoring logic are not directly compatible with real-time processing devices; thus real-time flight operational programs cannot be interfaced without some modifications to the operational flight code. These codes are typically implemented as firmware, and, as such, inherit the aforementioned change procedure problems. Add to this the fact that on-board processing devices are, in general, comprised of highly specialized, custom-tailored, ruggedized hardware components which are more expensive and limited in quantity when compared with off-the-shelf commercial equipment. Reference 6 provides a description of five basic design alternatives for incorporating these functions into the simulator; namely,

> "1) Actual (or equivalent non-flight-qualified) onboard computer (OBC) hardware
> 2) Translator/compilers
> 3) Interpreters
> 4) Functional simulation
> 5) Emulation
>
> Except for the functional simulation approach, all of these techniques utilize operational flight software. Use of actual on-board computer flight software is often necessary to meet simulation fidelity requirements and avoid time delays inherent in functional simulation software development and re-verification processes. Software changes to on-board programs occur with very short notice. Therefore, the simulator software should be capable of being rapidly updated and re-verified, and any equipment or software required to expedite this operation must also be provided."

In summation, a complete multiple-processor configuration design should include the delineation of any and all processor-related options which will be utilized to develop, implement, operate, maintain, and up-grade the operational trainer computational subsystem. Any and all anticipated enhancements should be clearly specified as to the type of changes, responsible party, change procedures, and configuration/quality assurance procedures.

This section has emphasized the flight training simulator characteristics and multiple processor configuration selection consideration. The following sections address the computational configuration design evaluation with respect to multiple-processor performance features.

3. GENERIC MULTIPLE PROCESSOR MODEL

3.1 Multiple Processor Performance Terminology

TBE has enumerated various performance evaluation features which are applicable to multiple processor application design configurations. In order to define these features, Figure 5 is used to present a simplified schematic of the active components during a real-time flight training simulator operation. Note

the computational subsystem interfaces with ND training device stations and NI instructor stations. The computational subsystem employs a real-time program comprised of NT operating system and application tasks implemented to provide automatic computations, logic, and I/O for a specified set of training mix activities. The task reference/update NB distinct data blocks to collect/provide the real-time inputs from responses to the trainees and instructors.

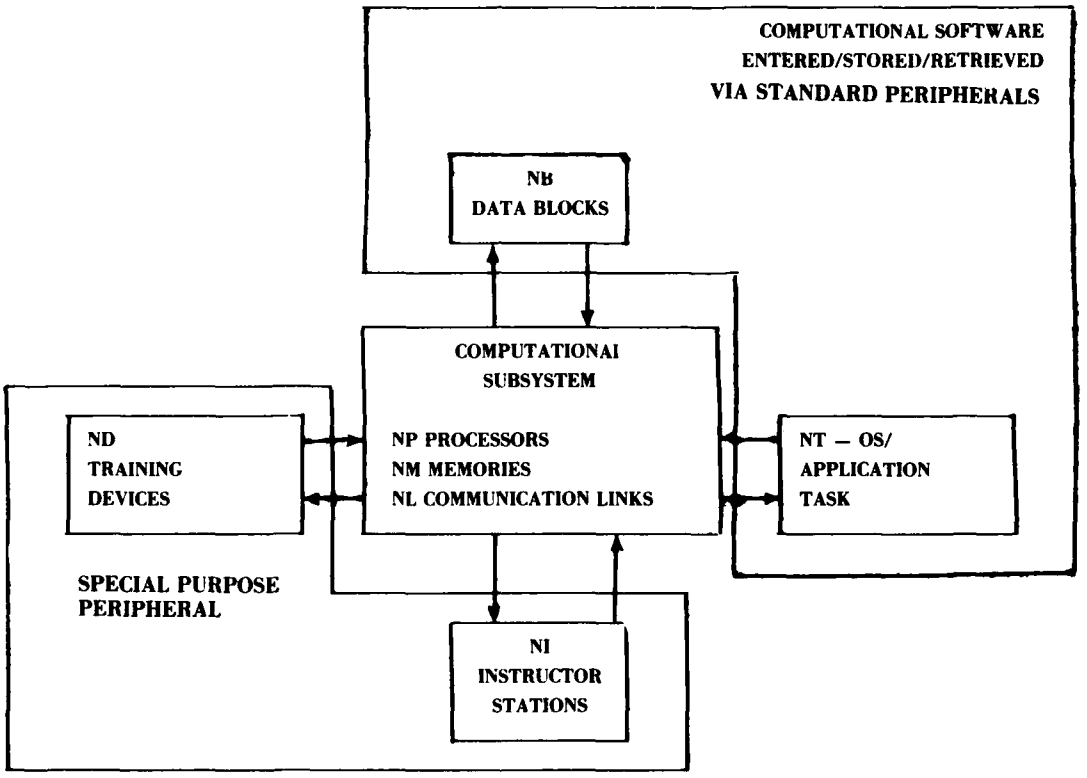


Figure 5. **Generic multiple processor configuration components for flight training simulation.**

The computational configuration hardware is basically a set of NP processors, NM memories, and NL communication links. The external peripheral devices may be categorized as general-purpose equipment versus special-purpose peripherals. Generals-purpose includes card readers, printers, magnetic tape drives, disc units, development/maintenance stations and other standard computer peripherals. Special-purpose refers to the training station and instructor stations devices to include controls, instruments, displays, visual imagery, motion, and "g" force equipment requiring computational subsystem monitoring, control, or support. It should be noted that the special-purpose devices may in fact be standard general-purpose devices such as a graphics CRT, but they represent a training application-peculiar interface. In some cases, a given device may be interchangeable in providing both training functions and general system operation/maintenance/development functions. This flexibility is one example of the built-in complexities which makes the separation of special application features from standard general vendor-supplied computation support features.

The generic types of computational interfaces with given types of peripherals assist in defining the following multiple processor performance terms:

1. Application load timeline
2. System throughput
3. Individual processor utilization
4. Individual memory/storage retrieval
5. Communication link traffic.

In addition, Figure 6 provides two extreme design alternatives with respect to multiple processor configurations which can influence the methods for representing and accounting for system benefits and penalties incurred as a result of specific application design. Flight simulators in general tend to combine these techniques as donoted in Figure 7. Another important feature to note is that a given trainee device station (D_1 or D_2 in Figure 7) is generally comprised of several different peripheral devices which may be serviced by different computational subsystems (for example, the visual, motion, and instrumentation subsystems). These features further complicate the traceability of a given trainee input to its incorporation as part of the simulated response. Each of the aforementioned performance-related terms is defined in the following subsections.

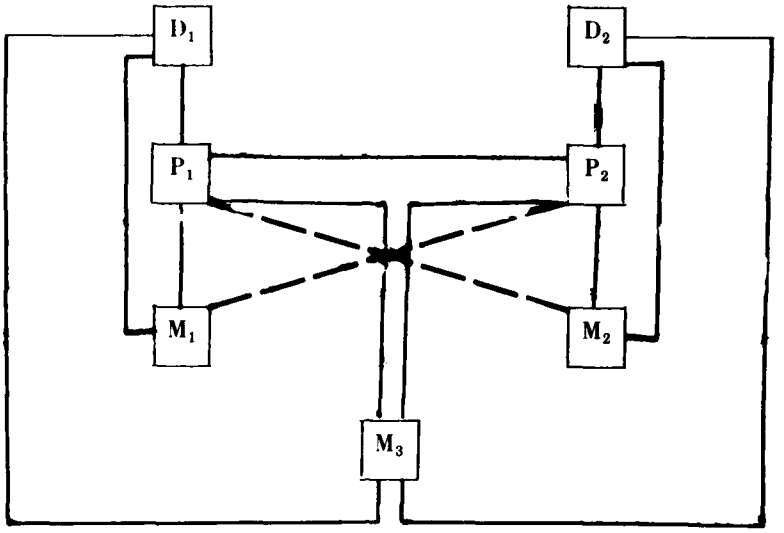
3.1.1 *Application Load Timeline.* In order to evaluate performance of any system, one must specify the application load timeline to be used as a performance measurement baseline. The fidelity of the baseline application load has a direct influence on the type of performance measurements which can be determined. For example, at a preliminary design level, a designer may wish to look at shared resource requirements, as a function of time-varied training mixes, to obtain a preliminary estimate of processing extremes and configuration tradeoffs at a functional level. At the system validation test point, actual test pilots, instructors, weapon operators or other appropriate "experts" may supply a planned real-time interactive baseline load for system performance evaluation. Regardless of the level of performance evaluation, it is important that a baseline set of load specification parameters and conditions, plus the performance measurements sought, be defined prior to delineating detailed means for the performance evaluation. This subsection delineates load parameters which characterize the flight training simulator computational load at various points during the system life cycle.

To properly represent a baseline load, one is interested in identification of the simultaneous training activities which are required to be serviced by the computational subsystem at any one time. The baseline source for this loading information is best maintained as part of the training facility specification. As a minimum, critical loading factors include:

1. The types of aircraft to be modelled
2. On-board crew stations to be modelled, including controls, instrumentation, visual, motion, and "g" equipments
3. Training mix of standalone positional training versus coordinated multiple crew configurations to be supported in terms of general facility training scenarios and maximum number of aircraft/positions involved.
4. Automated training guides to be keyed in order to control, perform, monitor, and score training scenarios defined in c.

From these required capabilities, the computational system engineers establish a timing and sizing budget which is based upon and traceable to a set of mathematically/logically expressed computational subsystem requirements. As a minimum, the functional tasks should be characterized in terms of

DEDICATED LINES



VERSUS
SHARED LINES

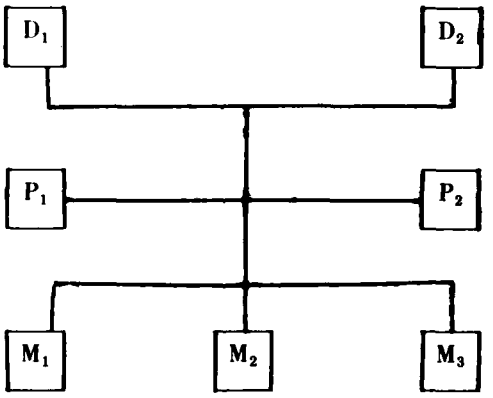


Figure 6. **Extreme configuration alternatives for processor/memory/peripheral device communications.**

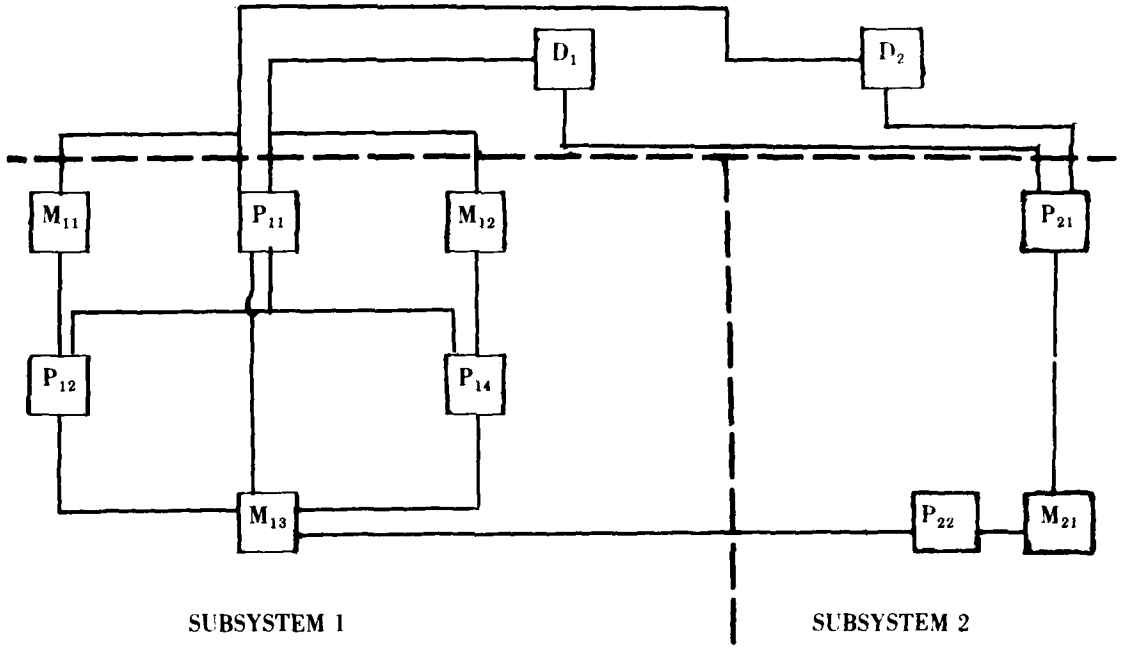


Figure 7. **Large flight training computational facility configurations incorporate combinations of dedicated and shared communication design.**

parameters listed in Table 2. Timing should be stated in terms of functional I/O rates, module iteration rates, and critical control path sequence (unconditional and conditional) coordination constraints of the mathematical model tasks. Sizing should address the I/O interface parameters and volume, temporary computational state data, and prerecorded data base structure and parameters necessary to support mathematical model computations. To further clarify the I/O interfaces, the application data blocks should be defined in terms of characteristics listed in Table 3.

During the design, the requirements timing and sizing budget provides one of the few measuring sticks as details are analyzed and allocated with respect to alternative candidate configurations. Because each simulator development organization has its own method for estimating the current design timing and sizing, it is very difficult to compare two competing designs. In most cases, the success or failure of a design does not become known until it has been coded and implemented on the hardware (which is generally a year to two years after the requirements are baselined and the development effort initiated). At this point, it is hard to determine if a failure is due to inadequater hardware, software, or if the timing requirement(s) are inappropriate. In other words, the system failure to provide a given training capability may be traceable to design and/or requirement problem areas. Tools are needed which can help identify design versus requirement-related bottlenecks and/or inconsistencies. Tools which can measure or verify current design parameters with respect to given requirements measures would fulfill part of this need. This requires that certain aspects of hardware/software design be standardized to trace application load requirements to the design components which are to handle the load. To better understand the application load, the other performance measures identified in 3.1 must first be considered in terms of the load environment.

Table 2. **Task Load Characterization**

Attribute	Values	Unit/Meaning
Identifier	6-Character Mnemonic	Provides a unique identifier for cross-reference and labeling purposes
Source Language	10-Character Code	Must match entry in the master source language list maintained for current processor technology
Instruction Mix for Each Instruction Type:		
• Instruction Identifier	10-Character Code	Must match entry in master simulator instruction mix identifiers
• Sizing Count	Positive Integer	Number of times this instruction appears in code
• Execution Count		Number of instruction interactions considering looping conditions for average and worst-case logic
Average	Positive Integer	
Worst Case	Positive Integer	
Data Retrieval for Each Task Input		
• Block Identifier	6 Characters	See Table 3
• When	6-Characters Code	
	= 'START'	All records read at first of task before main processing
	= 'ALONG'	Records processed one at a time
• Average Input	Positive Integer	Records
• Minimum Input	Non-Negative Integer	Records
• Maximum Input	Positive Integer	Records
Data Storage for Each Task Output:		
• Block Level	1 Character	See Table 3
• Block Identifier	6 Characters	See Table 3
• When	6-Character Code	
	= 'ALONG'	Records are output via individual processing
	= 'END'	Records are output just prior to task exit
• Average Output	Positive Integer	Records
• Minimum Output	Non-Negative Integer	Records
• Maximum Output	Positive Integer	Records
Enablement		
• Type	4-Character Code	
	= 'TIME'	Time Enabled
	= 'DATA'	Data Enabled
	= 'SLVD'	Slaved to Master Task
	= 'TAD'	Time and/or Data Enabled
• Frequency 1	Real	Interations/Second for Time Enablement
• Frequency 2	Real	Interations/Second for Data Enablement
• Frequency 3	Real	Interations/Second for Slaved

Table 3. **Data Load Characterization**

Attribute	Values	Unit/Meaning
Identifier	6-Character Mnemonic	Provides a unique identifier for cross-reference and labeling purposes
Level	1 Character	
	= 'S'	System Interface
	= 'G'	Global (used by more than one task)
	= 'L'	Local to one task but must be saved
	= 'T'	Temporary scratch area for a given task
Discipline	4-Character Code	Provides basic I/O requirement for determining suitable memory device allocation
	= 'FIFO'	Queue
	= 'LIFO'	Stack
	= 'SEQ'	Sequential
	= 'RAN'	Random
	= 'ROR'	Ready-Only Random
	= 'ROS'	Ready-Only Sequential
	= 'CBUF'	Circular Buffer
Sizing		
• Maximum Records	Positive Integer	Records
• Bits/Character	Positive Integer	Bits
• Characters/Word	Positive Integer	Bytes
• Average Words/Record	Positive Integer	Words
• Maximum Words/Record	Positive Integer	Words
• Minimum Words/Record	Positive Integer	Words

3.1.2 *System Throughput.* When describing a computational configuration, one frequently relates to the throughput capacity of the system. Data throughput is measured in terms of bits or bytes per second being transferred into or from the system. Processing throughput is measured in instructions per second (IPS) with typical reference being in terms of thousand (KIPS) or millions (MIPS) of instructions per second. In the case of a multiple processor configuration, the maximum throughput assumes all processors are busy and that the maximum I/O configuration capacity is active.

In designing a system, one rarely achieves or even desires maximum throughput. For one, a system operating at maximum capacity leaves no spare processing or I/O capabilities for potentially heavier loads. For another, it is a well-known observation from queing theory that a system which exceeds much more than 80% capacity is very susceptible to excess operating system overhead thrashing or shared resource bottlenecks. Thus, a design for a given application should (as observed in Section 2.3.4) incorporate sufficient spare capacity and growth.

As denoted in Section 2.3.4, spare capacity must be available at each of the basic configuration component levels (i.e. processor unit, memory module, or I/O communication link). Growth is the ability to add more components to the configuration. It should be noted that growth features provide more raw capacity but do not necessarily increase application throughout unless some design and/or match model changes are also made. This is particularly true for the case where a given sequence of dependent tasks are

the limiting factor in meeting a specific I/O port-to-port timing response. In this situation the task sequence represents a thread of activities which is sequentially activated by a given input port at, say, time t and is to compute a response needed for a given output port at time t + Δt. By definition, the thread activity is a sequential process, which negates simultaneous use of parallel multiple processors in configurations comprised of homogeneous processing elements. However, certain parts of the sequence may be best characterized by selected special-purpose functions which, if properly matched, may execute more efficiently by incorporating either special-purpose processors and/or firmware extensions to augment a general-purpose processor. Care should be taken to keep the design solution simple and to maintain a minimum number of well-defined yet flexible special-purpose interfaces.

The real-time task path data dependencies tend to be more complex than a set of parallel unrelated threads. Figure 8 represents a simplified set of the task/data relationships extracted from preliminary design (March 1979) of the visual general-purpose computational subsystem of the Advanced Simulator for Pilot Training (ASPT). Figure 8 illustrates the interfaces with the special-purpose visual computer image generation equipment (AOBJL, MODPLTST, DYNDATA, PAOL, DIRLITE, CAIPT, AOUT), aircraft math model position updates (RAWPAS and/or SIMPOS) and visual control panel consoles for two cockpits referenced as "A" and "B." The taks of Figure 8 represent the process relationships to be handled via CPUA and CPUB activated via a real-time 30-hertz interrupt pulse. In this particular example, the detailed output (AOBJL, MODPLIST, and DYNDATA) from the previous iteration is being output to the special-purpose display generator while the new position information is being used to determine outputs for the next iteration. The mathematics incorporated to integrate the position inputs typically require several iterations. Thus, an input at time t influences the output through time t + nΔt, where n represents the number of frames which constitute the number of positional inputs used to compute any part of the visual display. In this instance, the port-to-port time can be reinterpreted as the time required to completely execute n real-time frames. Each frame must be executed in less than a specified Δt seconds (0.033 second in example) to meet the math model requirements.

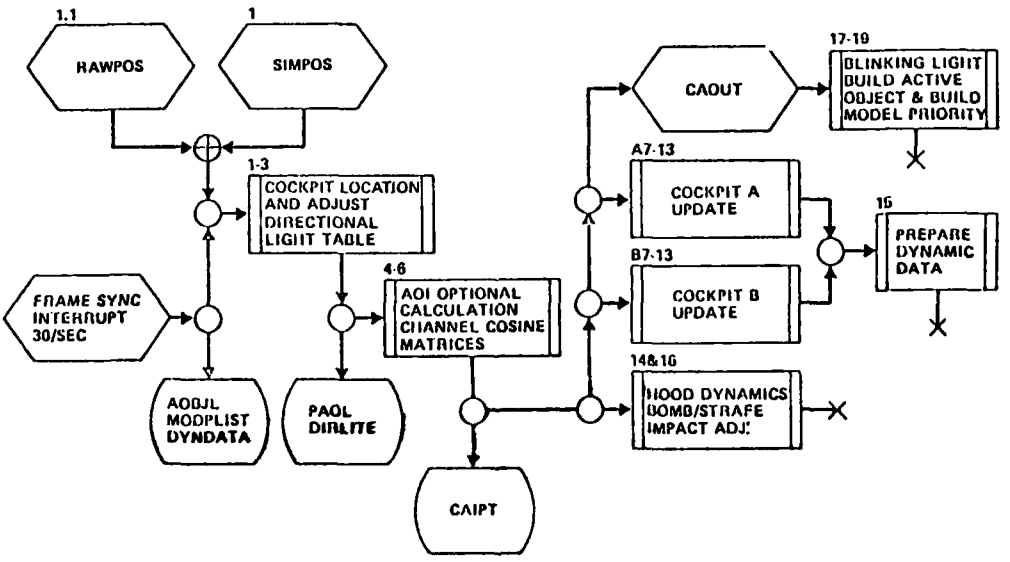


Figure 8. **Real-time application task flow.**

In the evolution of real-time flight simulator designs, the basic mathematical models have been modularized into parallel tasks requiring specified iteration rates. These tasks are generally characterized by a real-time application cycle (generally a one-second interval) which is subdivided into frames characterized by iteration rates such as 1/second, 4/second, 10/second and 20/second frames. The allocation of these tasks to a multiple processor configuration is an optimization problem which has been addressed in the Software Partitioning Study (Reference 7). For purposes of this study, it is sufficient to state that for a given real-time task partition allocation of the math model to a multiple processor configuration, certain assessments can be made regarding application throughput and/or bottlenecks. The supporting design data and techniques necessary to predict system application throughput are defined in Section 4 as part of the generic multiple processor performance prediction simulator.

In general, the maximum system throughput analysis requires detailed analysis of the specific application math model and its design alternatives. In most cases, time permits only a few alternatives to be considered. As long as the design timing and sizing measurement estimates are in tolerance, the designer tends to expand a given alternative being pursued. If a bottleneck is found or suspected, it is generally characterized as either being compute bound, memory bound, or an I/O bound problem. This leads to the necessity of individual component analysis throughout the design effort. If additional or replacement processing components cannot solve the bottleneck, then either a further segmentation of the math model or a compromise in iteration rates is required.

3.1.3 *Individual Processor Utilization.* A processor unit performs or executes based upon a stored program (whether it be a micro, mini, or mainframe based). The stored program for early 1960 flight training simulators was totally customed to meet the challenging real-time processing constraints of flight simulation. Generally a super general-purpose digital mainframe performed the mathematical model logic serving D/A and A/D customized interfaces with the cockpit training station instrumentation. As additional subsystems of motion, force, and visual aids were added, multiple processor, general-purpose combined with special-purpose equipment required further refinement of mathematical representation and computational requirements. These enhancements quickly saturated the processors for which the original flight simulations had been employed.

Meanwhile, hardware technology proliferated in areas of real-time control, command, and communication necessary to support various distributed processing configurations. In addition, software design technologies of top-down modular structure format provided many new design alternatives for processor utilization. This design freedom has proliferated instruction set architecture, as well as software development languages and support tools. One major fallout of this was the fact that independent detail design of the application can influence the structure of the real-time operating system necessary for managing processor resource utilization schemes. In particular, the processor rules for generalized multitasking (as opposed to customed application executives) have been expanded in the vendor-supplied, real-time operating system features. These schemes vary slightly from one processor to another, but in general address real-time critical task scheduling problems of priorities, sequences, data-triggered, and time-triggered activities. Many elaborate schemes have been devised; however, a few simple rules are best applied in the real-time flight simulation environment due to its cyclic tendencies.

First of all, there are two major sets of multiple processor design philosophies for real-time processor task assignments:

1. Centralized master processor with slaved processors whereby the master processor controls the task allocation and keeps track of the real-time cycle iteration counters assigning tasks to available processors. The slave processors have a very basic set of operating system rules which are driven by data set by the master processor.

2. Distributed task control operates based on a predefined set of processor tasks and processor task selection rules. This means each processor has a certain amount of executive process control logic to coordinate and interface its assigned tasks, and data with shared memory or I/O devices with explicit processor interfaces to keep the total process in synchronization.

Regardless of the processor control philosophy selected, certain vendor-supplied, real-time processor tasks controls can help streamline customed executives. These include:

1. Task priority service level assignments
2. Task enablement criteria
   - Time enabled
   - Data enabled
   - Interrupt enabled
3. Task activation within a priority loop
4. Task activation among priority loops.

These concepts have led to a straightforward table initialization/table driven real-time task manager which is less dependent upon the specific application design, and hence easily modified via changes to control table inputs which reflect a given design. The types of tables and logic schemes to carry out the real-time task process controls differ among vendors and even within vendor product lines. Thus, an independent modular computational application design, when mapped to specific configuration of hardware, should account for associated real-time vendor support software which is best suited to the design situation at hand. Automated techniques for matching design needs with vendor capabilities require parameter standards for both design traits and vendor products.

Once a task allocation scheme is selected, the individual processor timeline may be evaluated via static and dynamic analysis tools. Static tools operate on a given point-in-time set of constraints, such as a projected worst case set of constraints. Statis analysis generally concerns analysis of a given single frame or cycle interation. Dynamic analysis covers a broader investigation of the design relationships and operating rules to follow a timeline history of interactive cycles employing statistical and simulation techniques described in Section 3.3. At the processor level, the parameters of interest are shared memory access, computational speed of responses, and effectiveness/inadequacies of processor throughput with respect to iteration rates, processor spare/idle time, and instruction/data access and communication spare. Table 4 summarizes processor characteristics which influence performance evaluation of a given design.

3.1.3 *Memory Units.* Memory units, for purposes of this study, are used for both application tasks instruction and data. Processor technology has defined many levels of physical memory storage devices, core, metal oxide semiconductor, magnetic bubbles, disc, and tape, to name a few. The timing of a task is directly influenced by the physical type of memory accesses it makes. Current processor addressing schemes have facilitated the ability to "map" physical memories to several processors within a configuration via multiport memories. The memory controller itself is typically a microprocessor which identifies the processor requesting I/O access and prepares to ship or receive data. If multiple access occurs simultaneously, the memory handler determines in which order the requests are to be serviced.

As with processor units, memory I/O performance features may be determined by a static point-in-time analysis or a dynamic representation of the real-time application memory event time-line. Table 5 identified memory characteristics which influence performance.

Table 4. **Processor Characterization**

Attribute	Values	Unit/Meaning
Identifier	10-Character Mnemonic	Unique identifier for processor with the following attributes
Operating System		
• Multitasking		
▲Levels	Integer .GE.1	
▲Number of Priority Levels	Integer .GE.0 .LE. Levels	These many levels are serviced in a priority fashion. The remaining levels are serviced in a circular time-shared fashion.
• Enablements	Interger	Enablements/Second
▲Maximum Time Enablement Frequency		
▲Resource Management per Time Enablement	F10.9.GE.0	Microseconds
▲Maximum Data Enablement Frequency	Integer	Enablements/Second
▲Resource Management per Data Enablement	F10.9.GE.0	Microseconds
▲Maximum Slaved Enablement Frequency	Integer	Enablements/Second
▲Resource Management per Slaved Enablement	F10.9.GE.0	Microseconds
• For Each Task Level L		
▲Maximum Number of Task Level L	Integer .GE.1	
▲Task Service Scheme for Level L	Code	
	= 'P'	Priority
	= 'C'	Circular
	= 'F'	First-in, First Out
• Level Resource Management	F10.9 .GE.0	Microseconds
Simulation Instruction Set Measurement for Each Benchmark Instruction I		
• Sizing Measurements		
▲Number of Code Memories Involved		
The Memory Type for Each Code Memory m (the first memory is the user task code — any other memories are predefined for this processor)	4-Character Code	Must agree with master memory types defined in Table 5
▲Length of Code in Memory *m*	Integer .GE.1	Number of basic units used to describe memory *m* (see Table 5)
• Timing Measurement for Each Code Memory m and k=1,2		k=1 Implies Average k=2 Implies Worst Case
▲Number of Scratch Data Store Waits	Integer .GE.0	
▲Computational Total for All Memories	Integer .GE.0	Cycles
• Application Development Measurements Using Language L of the Master Language List		
▲One Item Development Charge	Integer	Man-hours
▲Change per Application Instruction of this Type	Integer	Man-hours

Table 5. **Memory Device Characterization**

Attribute	Values	Unit/Meaning
Identifier	10-Character Mnemonic	Provides a unique identification for each memory device in the technology data base for which the following attributes define
Type	4 Characters	
	= 'ROM'	Read Only Memory
	= 'RAMM'	Random Access Main Memory
	= 'RRAM'	Rotating Random Access Memory
	= 'SM'	Sequential Memory
	= 'WCS'	Writable Control Store
Size in Bits		
• Minimum	Positive Integer	Bits
• Maximum	Positive Integer	Bits
• Increments	Positive Integer	Bits
Number of Different Addressable Units	Positive Integer	
For Each Addressable Unit		
• Level	4-Character Code	
	= 'BIT'	Bit Addressable
	= '6BB'	6-Bit Byte Addressable
	= '8BB'	8-Bit Byte Addressable
	= 'WORD'	Word Addressable
• Bits/Unit Level	Positive Integer	Exclusive of Parity or Error Deletion Correction Bits
• Read Access Time	Real	Nanoseconds
• Read Cycle Time Unit	Real	Nanoseconds
• Maximum Sequential Units Transferred for Single Read	Positive Integer	Same as Unit Level
• Write Access Time	Real	Nanoseconds
• Write Cycle Time/Unit	Real	Nanoseconds
• Maximum Sequential Units for Single Write Access	Positive Integer	Same as Unit Level
• Error Detection/Correction	6-Character Code	
	= 'PARITY'	Parity Bit
	= 'SECDED'	Single Bit Error Correction Double Bit Error Detection
Number of Suppliers for Each Supplier—	Positive Integer	
• Identifier	10 Characters	Unique Identifier
• MTBF	Real	Hours — Mean Time Between Failures
• MTTR	Real	Hours — Mean Time to Repair
• MSPM	Real	Hours — Rescheduled Preventive Maintenance
• MTPM	Real	Hours — Mean Time for Preventive Maintenance

3.1.5 *Communication Links.* The utilization of shared memories among multiple processors requires shared communication to facilitate a variety of real time application needs. Some of the communication features which need to be considered are:

1. What is the maximum number of processor and memories which can effectively utilize a shared communications line?
2. How much built-in communication redundancy is necessary from an operationally reliable viewpoint?
3. What alternatives exist for identifying and quickly permit reconfiguring of a system to bypass a defective configuration component?

The answers to these questions, directly influence the selection of dedicated, shared, and backup communication devices incorporated into flight training simulator configuration designs. One of the major performance evaluation areas of concern is "What potential training modes will the candidate design support in degraded modes (i.e., loss of a processor/memory/peripheral/combine which requires troubleshooting maintenance and or repair)?" The candidate configurations analyzed tend to have preassigned processor-memory allocations which become hardcoded in their realization. This makes it very difficult to reconfigure the software, as well as physical hardware communication, unless it was an original design contigency requirement to be able to reconfigure under certain conditions.

As referenced in Figures 6 and 7, a wide variety of candidate design alternatives range from dedicated 2-way to multiple n-way communication configurations. The flight training computational subsystem must be able to externally service asynchronous and synchronous devices of varied speeds and buffer lengths. Internally, high-speed parallel data transfer devices are currently employed to ship burst transmissions of parameters from one computational subsystem to another. For example, the shipping of aircraft math model position data to visual image generation computations is done using a high-speed data, IISD, processor-to-processor transfer in ASPT. Within a given computational subsystem, a multiprocessor bus is commonly used to communicate with shared memory and to provide processor-to-processor communication.

The communication area has been identified as a major area for further study. For purpose of this study, the communication models for a candidate configuration are defined in terms of interfacing processors, memories, and/or peripherals with a maximum I/O rate which cannot be exceeded. Tools for communication reliability and reconfiguration analysis will require additional study to identify appropriate parameters and promising new technologies. Thus, the goals and recommended tools of this study assume fully operational communication configuration which are defined as part of the candidate design configuration being evaluated.

3.2 Goals

The goals selected for the multiple processor performance model were oriented toward timing, utilization, and initial resource predictions based upon given candidate configuration design specification (which includes software allocations) under an anticipated training environment load which incorporates features expressed in 3.1. The goals established are as follows:

1. Predict processor utilization statistics with respect to the real-time cycle to include:
   - Application task timing to include computational, I/O wait, and suspended wait time statistics
   - Resource management overhead timing
   - Spare time.

2. Predict communications traffic queing statistics with respect to real-time cycle to include busy, idle, and wait times for:
   - Channel utilization
   - Interprocessor transfers
   - Memory/processor transfers
   - Cycle and thread port-to-port timing statistics

3. From the statistical predictions, determine the following critical design resources with respect to unsatisfied timing and/or spare growth requirements:
   - Critical path task/data sequence
   - Potential bottlenecks

4. Provide summaries of selected candidate configuration utilization, queing, and critical resource statistics for which goals 1, 2, and 3 have been determined. Section 4.2 provides formats for specific report summaries in Figures 4.2 through 4.7.

5. For selected alternative candidate configurations, provide side-by-side summaries of selected statistics from goals 1, 2, and 3.

In establishing these design goals TBE considered a wide variety of *performance measurement goals*. The basic assumptions and feasible sample problem characteristics are now presented to provide further insight to the selection of this specific set of goals.

3.2.1 *Basic Goal Selection Assumptions.* Total system performance evaluation is an evolutionary, iterative processure which begins with conceptually desired training capabilities and ends (if successful) with an acceptable operational training device. The computational design provides a detailed layout of both hardware and software components in terms of their interrelationships and detailed process task flow (parallel and serial), data dependencies, and allocation of resources to service the required computational system I/O interfaces. The first assumption is that a baseline computational subsystem requirements model provides a mathematical and logical translation of inputs and associated computations to provide outputs along with associated response times to be present when system becomes operational. The second assumption is that the design performance can be measured with respect to these requirements given an appropriate set of measuring tools utilizing the selected design goals presented in this section.

These assumptions permit performance evaluation and prediction with respect to quantifiable requirements. Flight training simulator performance also incorporates human factors which deal with subjective measures of perceptions, emotions, and sensations. These types of performance measures require a prototype operational device with controlled statistical experiments of actual aircraft and simulated aircraft operations. Several extensive R&D facilities exist (in the Air Force and the National Aeronautics and Space Administration) which provide powerful computational, visual, motion, queing, aircraft models, and crew station model tools for testing and evaluating new algorithms for assessing human factors associated with design of training simulators and/or operational aircraft environments. Thus, a third assumption in selecting the multiple processor design evaluation goals is that the human factors have been studied and were incorporated as part of the quantifiable computational requirements or that actual prototype development will be required to address these issues.

These assumptions permitted the tools and techniques designed by this study to address specifically the multiple processor configuration design features in terms of application timing, sizing, throughputs and thereby the means for identifying any potential design bottlenecks, and provide for evaluating alternative candidate designs prior to prototype development and testing. These tools can also provide a means for evaluation of proposed modifications or additions to existing operational flight training simulation computational configurations.

3.2.2 *Sample Problem Characteristics.* The basic tools and techniques selected to satisfy the goals must be responsive to problem areas of candidate flight training simulator design configuration evaluation. For this purpose, TBE had defined sample problem features which were to be incorporated into the feasibility discussion of automated design evaluation tool of Section 4. A manual demonstration necessitates a simplified model of a real-time multiple processor configuration design.

The sample problem is described in terms of a generic multiple processor configuration depicted in Figure 9. This configuration supports two training device stations (D_1 and D_2) via external computational subsystem interfaces E_1, E_2, E_3, E_4, E_5, E_6, E_7, and E_8. Processors P_1, P_2, and P_3 represent a general-purpose mini-computer multiprocessor system with a shared data bus I_1, and shared memory M_2 and private memories M_1 *and* M_3 *for* P_2 and P_3, respectively. A high-speed data link I_2 interfaces the master processor P_1 with a special purpose computer P_4. The special-purpose computer has a dedicated memory M_4 which is accessible via communication interface I_5.

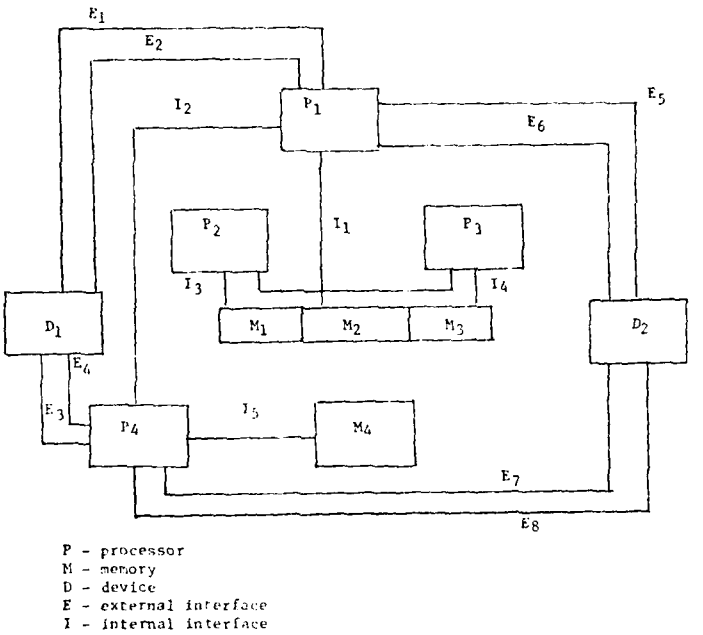


Figure 9. **Sample problem multiple processor configuration reflects shared/private memory, processor-to-processor, and external device *communications.***

To represent the process flow, the basic computational tasks relationships denoted in Figure 10 are used. T1 and T5 represent control coordination tasks. Tasks T2, T3, T4, T6, T7, and T8 represent tasks for device D. Similarly tasks T2′, T3′, T4′, T6′, T7′, and T8′ represent those same tasks for device D_2. Two iteration rates (30 hertz, 1 hertz) are required to meet the math model stability requirements. To complete the computational tasks flow description, the task data and instruction block requirements must be defined as described in Section 4.3.2.

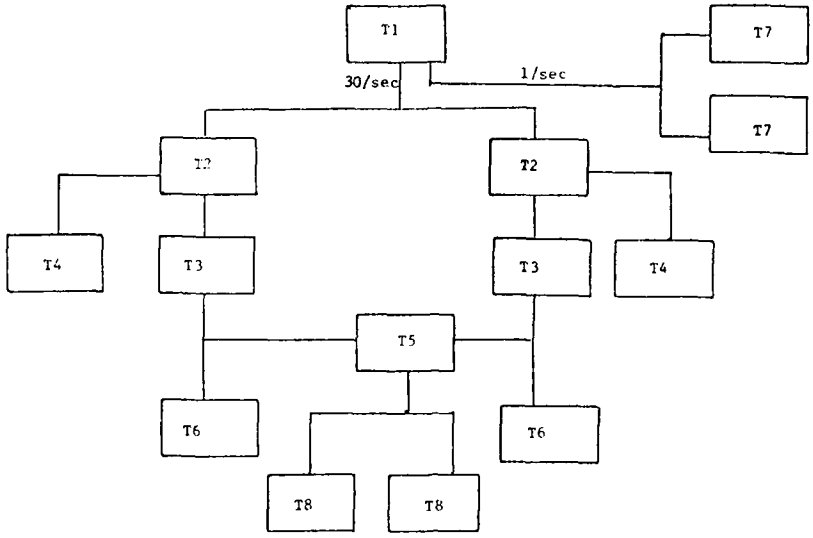


Figure 10. **Sample task flow indicates control task 1 and 5 with duplicate task (i.e., 2 and 2', etc.) to support the two training positions. Iteration rates have also been denoted as a fast loop (30/sec) and a slow loop (1/sec).**

A primary allocation of tasks to the candidate configuration can be defined as presented in Section 4.3.2. This allocation can then be perturbed when necessary to study alternative design features. At this point, we are ready to discuss the class of feasible tools and techniques identified during this study for evaluation of alternative candidate multiple processor designs.

3.3 Feasible Tools and Techniques

Complex systems such as flight simulators and other modern information systems require important planning decisions well in advance of actual implementation in order to assure that the system being planned is feasible and will attain its performance goals within the available resources. Alternative candidate approaches require evaluation as a part of the overall process of making design decisions regarding cost/performance tradeoffs.

Automated performance prediction programs are frequently called simulators or simulations; however, it is clear that these programs are distinct from the software involved in complete flight simulators. As shown in the Table 6, automated performance prediction programs include several classes of models, with each class involving various levels of complexity and having various primary applications. We have found that there is frequently a need to develop models that include some features of various simulation classes in various parts of the model in order to tailor the tool to the issue at hand.

The tool categories presented in Table 6 are ordered from the most straightforward to highly sophisticated, special algorithm-analysis tools. In most cases, the outputs from the more detailed tools can

Table 6 Categories of Design Performance Prediction Tools

Category	Application Analysis
Static Resource Models	Estimated timing and sizing predictions for processor utilization, memory utilization, and network data rates based on computational executive real-time cycle incorporating iteration rates and functional task timing and sizing data.
Functional Simulations	Modelling the major system I/O loading and control logic for analysis and prediction of CPU utilization, functional allocation, data flow, queuing delays, resource usage, resource allocation, growth and degradation, and overall system organization.
Analytic Simulations	Mathematical representation of a specific algorithm for studying their accuracy, detailed logic, and computational tradeoffs.
Emulators	Special software and/or hardware which produces exact computational results, playing the role of specialized hardware that is currently unavailable, to execute or emulate an algorithm represented in object code format for proposed specialized hardware.

be reduced to provide refined inputs to the simplified tools. The ease or difficulty of using these tools in a correlated fashion is directly related to the extent with which a global data base design and tool I/O interface definitions are coordinated. This section emphasizes the distinction among and definition of each of these analysis levels.

Table 7 relates these tools to the design goals established for multiple processor candidate design evaluation techniques. The static tools can provide a single value estimate for most of the measurement goals. Static tools are not able to adequately assess dynamic measures such as suspended wait time, port-to-port cycle times, and critical paths without some sort of interactive interface and knowledge of a functional or analytic model. The functional tools can provide statistical estimates for all the basic measurement goals. Their validity is subject to the fidelity of the inputs and level of computational models utilized. Analytic and emulation tools generally address a subset of the computational subsystem in terms of a specific task algorithm. For this reason they provide detailed insight as to computational accuracy and timing of the given algorithm but would be too voluminous and unwieldy to address the total computational subsystem statistics. Regardless of the tool being utilized, a well-defined design data base and report generator can greatly simplify the user interface and coordinated use of multiple tools in performing candidate configuration evaluations.

The feasibility of each tool directly relates to the data collection, automated procedures, configuration management, and quality assurance provisions for applying the tools. As with any tool, the quality of the input and the fidelity of the model directly influence the interpretation of and confidence in the output. One of the feasibility issues concerns the number of runs required to obtain meaningful answers. Directly related to the number of runs are the input data and parametric controls necessary to accommodate the runs to obtain a meaningful interpretation of outputs.

Table 7. **Tool Category Relationship with Design Goals**

Design Goals	Tool Category			
	Static*	Functional	Analytic**	Emulation**
1. Predict processor utilization statistics with respect to the real-time cycle to include:				
• Application task timing to include computational,	Yes	Yes	Yes	Detailed
I/O wait,	Yes	Yes	Partial	Partial
and suspended wait time statistics	No	Yes	Partial	Partial
• Resource management overhead timing	Yes	Yes	Partial	Partial
• Spare Time.	Yes	Yes	Partial	Partial
2. Predict communications traffic queuing statistics with respect to real-time cycle to include busy, idle and wait times for:				
• Channel utilization	Partial	Yes	Partial	No
• Interprocessor transfers	Yes	Yes	Partial	No
• Memory/processor transfers	Yes	Yes	Partial	Detailed
• Cycle and thread port-to-port timing statistics	No	Yes	Partial	No
3. From the statistical predictions determine the following critical design resources with respect to unsatisfied timeing and/or spare growth requirements:				
• Critical path task/data sequence	No	Yes	Partial	Partial
• Potential bottlenecks.	Yes	Yes	Yes	Yes
4. Provide summaries of selected candidate configuration utilization, queuing and critical resource statistics for which goals 1, 2, and 3 have been determined. Section 4.2 provides formats for specific report summaries in Figures 12 through 17.	**Data Base/Report Generator Required**			
5. For selected alternative candidate configuration provide side-by-side summaries of selected statistics from goals 1, 2, and 3.				

*Provide a single estimate as opposed to statistical estimate.

**Require multiple runs to obtain statistics

In our experience with actual performance prediction problems, we have found that various subsystems of the system need to be modeled to different degrees of accuracy. This means that some of the subsystem models must be functional models, and others must be analytic models or emulators. Proper design and implementation of a simulator provide a highly structured simulator with sufficient flexibility to allow tailoring each module to be functional, analytic, or mixed as required to support the overall objectives. This flexibility is important because of rapid increases in the cost of development and operation as more fidelity is demanded. The model must include only those aspects of the system that are relevant to the study objectives. Optimizing a model to a single system configuration is not desirable, however, since the lose in flexibility causes higher maintenance costs throughout the life of the

simulation. TBE's general approach to the design of the performance prediction model has included special attention to the cost, performance, and growth potential tradeoffs that are useful in the planning and evaluation of complex flight training computational subsystems.

Our design, presented in Section 4, focuses upon a generic multiple processor functional simulator which can interface via data base and manual selection with other tools. The related tools with which the recommended functional simulator may interact are now described with respect to the flight simulator multiple processor design evaluation role. These include static, analytic, and emulation tools.

3.3.1 *Static Resource Models.* Static resource models (sometimes called "beancounters") are typically straightforward programs that sum resources used as functions of inputs involving resources per execution and execution rates. Depending on the complexity of the system being modeled, these static resource models can range from almost trivial to quite complex. For example, TBE has developed and utilized a complex model in this class, called CERT, which accounted for the CPU resources used by a very sophisticated operating system for the Site Defense Ballistic Missile System.

TBE's Software Partition Study contract with AFHRL (Reference 7) provided the details of a static resource analysis model for flight training simulator computational design partitioning analysis. This analysis tools requires manual interaction techniques to establish a baseline partition of the computational tasks relationships which are independent of a specific computational hardware configuration. Once this baseline partition of the mathematical model has been translated to computational tasks and the data/timing relationship have been established, the partitioning tool facilitaties analysis and allocation tradeoffs for any given candidate hardware configuration. As pointed out in this related study, the allocation of a given computational partition is a mathematical optimization problem. The final report for the Software Partitioning Study documents the optimization model and a heuristic design for its implementation as an automated design analysis/evaluation tool.

A major emphasis for this tool is the development of a flight training computational design repository which would greatly facilitate timing and sizining estimates during the preliminary design phases of a new training simulator. In addition, the tool facilitates evaluation of proposed design changes and tradeoffs to an existing facility prior to change implementation. The data base structures defined by the partitioning tool provide a means for unambiguously expressing task/data flow relationships. It also provides for the separation of hardware technology timing and sizing attributes from the computational task descriptions via the establishment of a technology data base. Using this technique, a task is described in terms of a standardized macro instruction mix which can than be automatically translated to specific timing and sizing for a given processor/memory configuration.

The extensive partitioning data base combined with a processing allocation and the configuration communication priority schemes form the major inputs needed to drive a functional simulation of a generic multiple processor model. The processing allocation of tasks to processors and data blocks to memories is the major output of the Partitioning Algorithm. The interpretation of the configuration communication rules and processing priorities is the major emphasis of the functional simulation tool defined by this contract in Section 4. The two tools are part of an iterative design evaluation tradeoff process to predict candidate configuration performance.

3.3. *Analytic Simulation Models.* The term analytic simulator is typically used to describe simulators that contain mathematical computational algorithms. These simulators are usually used to study the accuracy and detailed logic behavior of an algorithm. For example, TBE has developed and utilized analytic simulators for detailed studies of tracking filter algorithms and radar scheduler algorithms. In general, analytic simulators do not model systemwide timing and queing considerations; however, load and timing considerations are important in selecting which algorithms to emphasize in the analysis. Analytic models usually require realistic input data. These simulators are most useful for studies requiring detailed computations.

In the case of flight training simulators, there are many numerical analysis tradeoff decisions which require a high-fidelity, non-real-time mathematical algorithm to study relationships of specific aircraft body and weapon systems math model configuration responses. Typically these analytic math models are enumerated in FORTRAN for large mainframe computers. They can provide a fairly efficient means for delineating or verifying new algorithm computational logic and accuracy.

In general, these models require detailed data for the specific aircraft/weapon system recorded performance measurements from controlled flight test environments. The baseline application load requires detailed initial state values, flight profile, and time-tagged weapon-mode activation switches which correlate with the flight profile. By careful parameterization and mathematical modularization, this analytic model can be used to refine estimates of selected computational module partitioning task parameters and provide statistical distributions for functional timing and queing sizes. The utilization of a non-real-time model permits concentrated analysis of the required high-level computational instruction mix, control sequence logic, and accuracy of a candidate algorithm. They play an important role in math model requirements in terms of determining real-time iteration as a function of integration step size as well as providing detail design hardware and language selection features.

3.3.3 *Emulators.* The term emulator usually refers to an instruction-level simulation of another computer in order to duplicate the processing of that computer. Emulation can be accomplished through software techniques or by using micro-code to program a microprocessor to interpret the instruction set of the target computer and duplicate the output of the processing initiated by a command. Emulators of this type are only used when extreme computational fidelity is required and where the actual software instructions are available.

Emulation is generally reserved for making tradeoffs in new hardware technology areas where the prototype hardware and support software do not currently exist. They may also be employed to determine the impact of a modified or extended instruction set prior to building a prototype. In the flight training computational subsystem evaluation, certain benchmark algorithms are selected for coding in the proposed machine language and then the code is emulated (executed) with diagnostic dumps, displays, and printouts to determine its performance in terms of timing, sizing, and accuracy. The benchmark algorithms are restricted to a class of task computational characteristics for which the proposed instruction architecture is reportedly well suited or designed to handle.

In many instances, the software and hardware necessary to support a flexible emulation user interface costs more than it would to build a prototype. This is particularly true in the evaluation of any radically different instruction set architecture. At the other extreme, the actual hardware may exist, but very little development and diagnostic support tools exist. (This is particularly true with microprocessor technology.) In this situation, a mini-computer may provide a full complement of cross-compilers, -assemblers, -link editors, and emulators for algorithm development and initial debugging prior to downloading the object code or burning the ROMs for the object computer. In this latter situation, emulation is a test tool rather than a design tool, although real-time critical new algorithm design could employ the same tools to pursue detailed tradeoffs as necessary.

In summary, a variety of automated techniques are not only feasible but have been applied in specialized design evaluation settings. The feasible adaptation of a general set of multiple processor design evaluation tools is directly linked to requirements and design specification standards to facilitate a quality-controlled automated data entry/reduction process. Static tools such as the Partitioning Algorithm provide systematic accountability of a diverse number of interrrelated requirements, design, hardware, and software parameters. Functional simulation tools exercise the operating rules and communication rules to carry out task sequences and data dependencies. Analytic and emulation tools are reserved for specific new algorithm mathematical and implementation tradeoffs, respectively. From these, the functional simulation was selected for detailed expansion as it best meet the design goals for a generic multiple processor performance model.

4. MODEL DESIGN

4.1 Model Overview

Figure 4.1 presents an overview of the computational subsystem performance predictor evaluation environment. The evaluator interfaces with a design repository and functional simulator to obtain outputs of predicted performance. The evaluation report outputs (Section 4.2) are the result of a segmented repository of flight training simulator candidate design description inputs (Section 4.3) combined with the discrete event generic computational subsystem functional simulation described in Section 4.4.

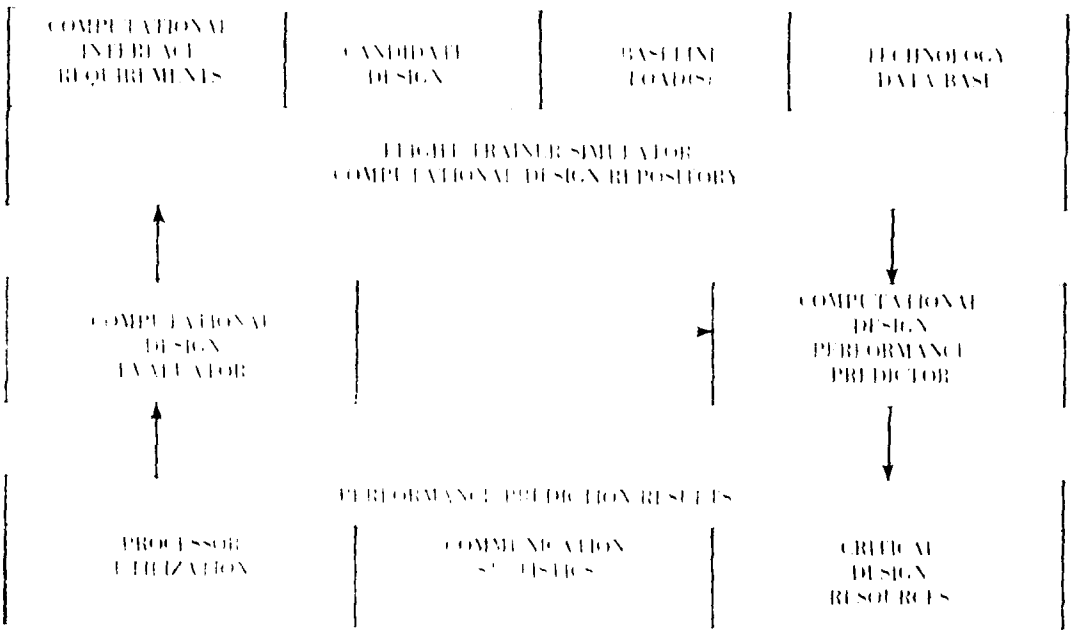


Figure 4.1 **Model overview depicts *interfaces* with the evaluator user, a segmented flight trainer computational design data base repository, and an evaluation statistical output data base.**

4.2 Model Outputs

The output performance measurements for computational candidate design evaluation are based upon individual configuration component performance and the system throughput response times for the given application evaluation. For purposes of this model, configuration components are defined to include processors, memories, and communication links which are netted together to service the external device loads. Each of these takes on two basic states at any given time, namely busy or idle. The basic output corresponding to this fact is a component summary of busy/idle measurements in terms of percentages and/or accumulative times. Figure 4.2 represents an output report format which permits saturated component utilization to be quickly identified via a flag set by the check of required spare growth percentages (on a component basis) to simulated utilization.

BASIC CANDIDATE DESIGN COMPONENT UTILIZATION

CATEGORY ID	COMPONENT ID	DEVICE ID	----TIME BUSY----- PERCENT	SECONDS	-----TIME IDLE----- PERCENT	SECONDS	REQUIRED SPARE PERCENT	FLAG
CC	CCCCCC	DDDDDDDDDD	99.999%	999.999999	99.999%	999.999999	99.999%	C

Figure 12. **Basic candidate design component utilization facilitates a quick look evaluation of spare utilization and provides a means for flagging critical design components near or above required growth factor for given simulation load.**

Once a major component is determined to be a problem area which requires additional analysis, more detailed component reports have been designed to help identify the bottlenecks. These reports are tailored to the component category as illustrated in Figures 13 and 14 for processor utilization and Figures 15 and 16 for memory utilization. It is important to note the statistical summary (in terms of number of times a given event occurred, such as an application I/O request, along with minimum, maximum, average, and standard deviation statistics for the particular measurement) provides dynamic range and variance of single measurements with respect to application resource requirement measurements such as task executions and data block storage/retrieval parameters.

PROCESSOR UTILIZATION

IDENTIFIER	DEVICE	APPLICATION	OVERHEAD	IDLE
\\\\\\	\\\\\\\\\\	999,999 %	999,999 %	999,999
		999,999 SEC	999,999 SEC	999,999 SEC
\\\\\\	\\\\\\\\\\	999,999 %	999,999 %	999,999 %
		999,999 SEC	999,999 SEC	999,999 SEC

Figure 13. **Processor utilization summary report is designed to give basic accountability for each processor in terms of application computations, resource management overhead, and idle time for the simulated period.**

PROCESSOR DETAILED TASK MIX FOR PROCESSOR PPPPPP DEVICE DDDDDDDDDD

TASK ID	EXECUTIONS STARTED COMPLETED	REQUIRED	TIMING STATISTICS PER COMPLETED EXECUTION IN MILLISEC ----COMPUTATIONAL---- MIN MAX	MEAN S DEV	---------I/O--------- MIN MAX	MEAN S DEV	------SUSPENDED------ MIN MAX	MEAN S DEV	----- TOTAL ----- MEAN MAX	FLAG
TTTTTT	999999	999.99999	999.999999	999.999999	999.999999	999.999999	999.999999	999.999999	999.999999	C
	999999		999.999999	999.999999	999.999999	999.999999	999.999999	999.999999	999.999999	C

Figure 14. **Processor utilization detailed task mix report is designed to present application task timing statistics plus number of executions started and/or completed for the simulated period.**

MEMORY UTILIZATION BY PROCESS I/O REQUEST

MEMORY DEVICE ID	PROCESSOR DEVICE ID	REQUEST TYPE	--SIZE IN UUUUUUUU OF 999 BITS-- -- PER REQUEST-- MIN / MEAN / MAX / S DEV	----TOTAL----	--SERVICES TIME IN MICRO SECONDS-- ------PER REQUEST------ MIN / MEAN / MAX / S DEV	--TOTAL--	BASIC ACCESS TIME (NSEC)
MMMMMM DDDDDDDDDD	PPPPPP DDDDDDDDDD	READ	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	9999.
		WRITE	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	9999.
		TOTAL	9999999 / 9999999 / 9999999 / 9999999	9999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	N/A
	ACCESS WAIT - READ			999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	
	ACCESS WAIT - WRITE			999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	9999.
MMMMMM DDDDDDDDDD	---------- ---TOTAL---	READ	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	9999.
		WRITE	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	9999.
		TOTAL	9999999 / 9999999 / 9999999 / 9999999	9999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999	N/A
	ACCESS WAIT - READ			999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	
	ACCESS WAIT - WRITE			999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	999999999.	

Figure 15. **Memory utilization by processor report permits identification of processor/memory access bottlenecks in terms of wait times and required service times.**

DATA BLOCK SUMMARY FOR MEMORY MMMMMM

DATA BLOCK	REQUEST TYPE	--SIZE IN UUUUUUUU OF 999 BITS--- -- PER REQUEST-- MIN / MEAN / MAX / S DEV	----TOTAL----	--SERVICES TIME IN MICRO SECONDS-- ------ PER REQUEST------ MIN / MEAN / MAX / S DEV	-- TOTAL--	-BLOCK LENGTH- --IN RECORDS-- MIN / MEAN / MAS / S DEV
BBBBBB	READ	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	9999999999.	99999 / 99999 / 99999 / 99999
	WRITE	9999999 / 9999999 / 9999999 / 9999999	999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	9999999999.	99999 / 99999 / 99999 / 99999
	TOTAL	9999999 / 9999999 / 9999999 / 9999999	9999999999	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	9999999999.	
ACCESS WAIT -	READ		999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	9999999999.	
ACCESS WAIT -	WRITE		999999 REQUESTS	99999.9999 / 99999.9999 / 99999.9999 / 99999.9999	9999999999.	99999 LOST

Figure 16. **Application data block summary permits sizing and storage/retrieval timing to be analyzed for potential design reorganization if critical wait time bottlenecks are detected.**

In summary component level measurements output by the model include the following:

1. Component busy/idle measurements for given evaluation time period.

2. Processor utilization in terms of resource management overhead, application task mix (computation, I/O, and suspend) services and idle periods.

3. Memory utilization in terms of processor communications and in terms of data block communications including types of accesses and number of accesses in addition to measures of service and wait times.

4. Communication link utilization parameters in terms of total traffic throughput and wait times by interfacing devices.

In addition to configuration components, external device I/O responses require that the system throughput for given inputs be traced via the model to determine associated response(s) times. In the real-time flight training environment, training device outputs are extrapolated based upon a combination of input samples, aircraft and weapon system models plus other environmental and equipment specific models. Thus, a single input source is seldom directly traceable to a given output response. Instead, the software design real-time cycle, subcycles and frames become the important synchronization of frame, sub-cycle, and cycle computational support completed/uncompleted. In particular sequentially dependent software task threads within each of these levels must meet appropriate timing requirements. The model is designed to track and summarize related task timing data in terms of busy versus idle statistics for four different iteration rate subcycles incorporated to define a real-time cycle. It also accumulates end-to-end timing statistics for 20 dependent task threads. A sample output format is illustrated in Figure 17. These dimensions can be increased given that adequate storage is available for the many variables used to track and accumulate the statistics.

APPLICATION THROUGHPUT

CYCLE/ THREAD	IDENTIFIER	-----REQUESTED----- LIMIT	SPARE	OCCURANCES OBSERVED	-END-TO-END-TIMING- MIN MAX	MEAN S DEV	FLAG WAM
CYCLE	CCCCCC	.999999 SEC	99.99%	999999.	9.999999 9.999999	9.999999 9.999999	FFF
THREAD	TTTTTT	.999999 SEC	99.99%	999999.	9.999999	9.999999	FFF

Figure 17. **Application throughput outputs are modelled for at least four levels of software iteration rates and 20 different threads to obtain end-to-end timing statistics for comparison with required time limits and built in spare.**

Tabular report outputs are not the only convenient means for consolidating and presenting the simulated performance parameters. Histograms can provide a visual comparison of parameters measured at given periodic intervals over the simulated evaluation time period. Figure 18 is an illustration of processor utilization. The types of reports and displays which can be generated are many; however, the basic output parameters remain the same. These parameters are defined in Appendix A.

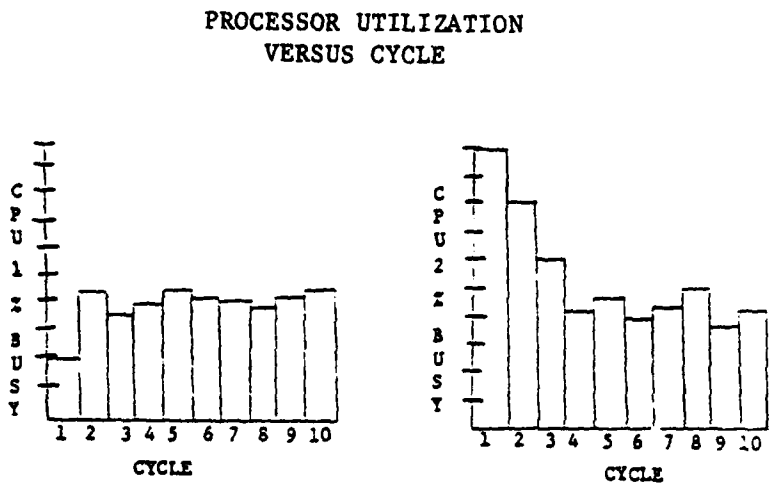


Figure 18. **Histograms of various parameters such as processor utilization provide a visual means of the load balance among processors and peak cycle or frame loading with respect to the simulated evaluation baseline load.**

4.3 Model Inputs

The amount of input parameters necessary to obtain the outputs is directly related to the selection of hardcoded models versus generic data driven models incorporated in a discrete event simulation model. Generic models by their very nature require more data, but, as a general rule, permit a wider range of user problems to be analyzed in a parametrically controlled data environment without changing the model code. On the other hand, a hardcoded analytic model may be a more precise representation of the given entity under study.

In the case of flight training simulator computation configuration evaluation, there is generally an abundance of data and design documentation descriptions which must be sorted out and reviewed for completeness of the design specification, as well as, ascertaining if the design will satisfy the training computational interface requirements. Therefore, candidate design evaluation is primarily analysis of supplied data permitting the feasibility of a generic evaluation model. The model has been designed to provide an organized method for extracting and entering the data to automate evaluation of the more complex relationships which must be satisfied by the candidate design.

In organizing the inputs, the following have been used as the major areas of data collection/entry.

1. Computational Interface Requirements
2. Candidate design
3. Baseline load
4. Technology data base
5. Evaluation options.

A brief observation, with regards to the software partitioning data base (Reference 7) and its relationship to the performance prediction model, is now given. The computational interface requirements are the same. The candidate design area contains the resultant software partitioning allocation (or a manual candidate design allocation if simulation is run independent of partitioning) of the baseline software task for a given candidate configuration. The baseline load is a combination of the static partitioning software task load parameters plus additional dynamic load distribution parameters. The technology data base remains the same. The evaluation model options are an expansion of the software partitioning evaluation parameters to permit parametric model controls and output report selection. Each input area is now addressed as it specifically applies o the computational simulation performance evaluation model. Specific input parameters for each of these five data base areas are listed in Appendix B.

4.3.1 *Computational Interface Requirements.* Computational interface requirements provide the inputs necessary to model the external device interfaces with the computational subsystem to be simulated. Definitive quantitative requirements provide the basic foundation for both system design and system modelling. As a minimum, all required external system I/O interfaces must be specified along with the system functions to be performed. Table B-1 of Appendix B presents the interface requirements inputs as three major groups of parameters; namely:

1. Group 1 — System Requirements Identifier provides unique identification of the system under consideration.

2. Group 2 — System Interface Requirements address the number of devices and required options such as I/O service frequencies, buffering parameters, etc. These parameters are correlated with the technology data base described in Section 4.3.4.

3. Group 3 — System Function Requirements address the number of system functions, specify function execution rate and timing requirements, maps system interfaces serviced, nd defines function to function interfaces.

The version of the algorithm designed under this contract utilizes system function requirement inputs as a manual step to obtain software load and task definitions of the candidate design. The system interface definitions provide the external sources that the candidate configuration must service. Future model expansions could provide an automated means for traceability between system level modelling and computational design detailed modelling.

Referring to the sample problem stated in Section 3.2, there are eight defined external interfaces E_1, E_2 . . . . E_8. Figure 19 represents the input form designed for external interface device definitions. Note that each device must be either a communication link CL, Processor Unit PU, or Memory NM.

4.3.2 *Candidate Design Allocation.* The design allocation parameters (Table B-3 of Appendix B) include the candidate multiple processor configuration definition with processor software task and memory data block assignments plus the static task/data block I/O partition relationships. This is essentially a consolidation of the baseline software tasks, candidate hardware configuration, and an allocated partition organized in the following groups as input to the model.

1. *Group 1 — Design Identifiers* permit traceability to specific data origination for this particular design allocation to include an indicator of manually generated versus software partition algorithm generation.

2. *Group 2 — Basic Design Parameters* provide for dynamic sizing of the simulation control tables and event attribute files for simulation model bookkeeping.

COMPUTATIONAL SUBSYSTEM INTERFACE REQUIREMENTS FILE IDENTIFIER

REQUIRED COMPONENTS

COMPONENT		UNIQUE SYSTEM IDENTIFIER	OPTION 1, 4, 7, ...	OPTION 2, 5, 8, ...	OPTION 3, 6, 9, ...	CONTINUE
TYPE	DEVICE					
PU	PROCESSOR DEVICE		1 AC[illegible] LEVELS 4 LA[illegible] E 2	2 OPERATING SYSTEM 5 LANGUAGE 3	3 LANGUAGE 1 6 - LANGUAGE 4	CONTINUE IF MORE LANGUAGES ARE SPECIFIED
MM	MEMORY DEVICE		1 - SIZING UNIT	2 - SIZE		
CL	COM DEVICE		INTERFACING DEVICE TYPE	INTERFACING COMPONENT	PRIORITY	CONTINUE IF MORE INTERFACING DEVICES

Figure 19. **Required components are characterized as processors, memories or communication lines. Communication lines may interface with external devices such as instructor stations, cockpit controls, cockpit instrumentation, standard computer peripherals, display subsystems, and others which have been defined in a technology data base.**

3. *Group 3 — Configuration Components* define the processors, memories, external devices nd communication links which comprise the multiple processor configuration to be modelled.

4. *Group 4 — Data Block Attributes and Allocation* provide the means for defining data set storage/retrieval discipline for processor/memory communication. The attributes supplied are correlated with technology data base storage definition parameters and memory assignment and sizing allocation to a given block of a memory storage device of the candidate configuration.

5. *Group 5 — Task Definitions and Allocated Processor(s)* provide detailed timing, sizing, I/O interfaces, I/O volume, and enablement criteria for each task including assigned processor(s) and instruction block(s) memory assignments. A major emphasis is definition of the design communication rules for tasking and priorities for handling competing processor, memory, and external devices I/O communications which will interface with the simulated load models.

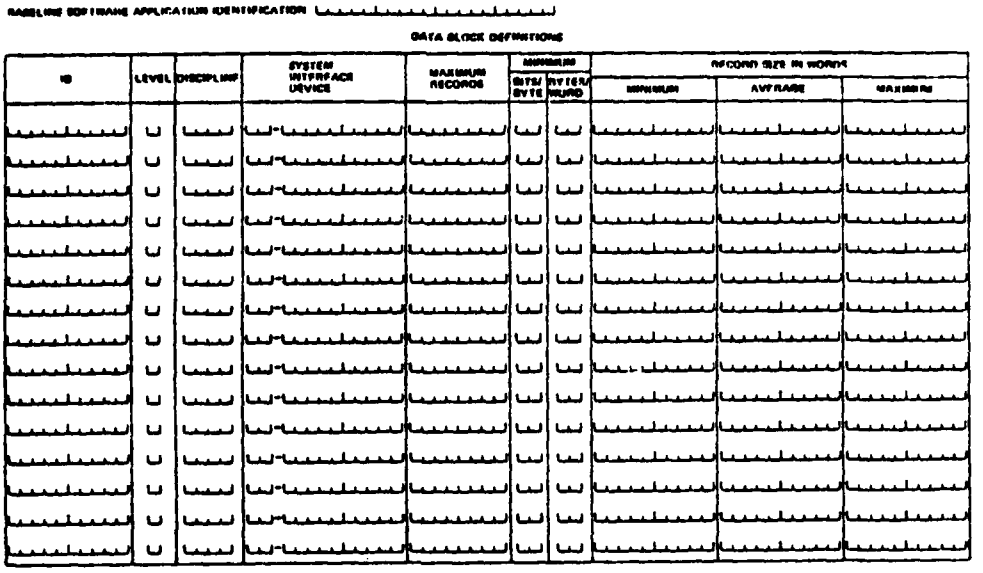
Four input forms (Figures 20 through 23) have been designed to provide a means for consolidating the manual recording and data entry process. It should be noted, that our flexible design incorporates automatic counters for the basic sizing parameters. These include the number of processors, memories, data blocks, instruction blocks, communication links, tasks, and other countable entries.

CANDIDATE CONFIGURATION IDENTIFIER

PROPOSED DEVICES TO COMPLIMENT REQUIRED DEVICES
LIST COMMUNICATION LINES LAST
REPEAT REQUIRED CORLINES IF ADDITIONAL DEVICES ARE INTERFACED

COMPLIMENT		CANDIDATE IDENTIFIER	OPTION 1, 4, 7, –	OPTION 2, 5, 8, –	OPTION 3, 6, 9, –	CONTINUE
TYPE	DEVICE					
PU	PROCESSOR DEVICE		1 - ACTIVE LEVELS 4 - LANGUAGE 2	2 - OPERATING SYSTEM 5 - LANGUAGE 3	3 - LANGUAGE 1 6 - LANGUAGE 4	CONTINUE IF MORE LANGUAGES ARE SPECIFIED
MU	MEMORY DEVICE		1 - SIZING UNIT	2 - SIZE		
CL	COM DEVICE		INTERFACING DEVICE TYPE	INTERFACING COMPLIMENT	PRIORITY	CONTINUE IF MORE INTERFACING DEVICES

Figure 20. **Configuration component definition is designed to be input the same as required components.**

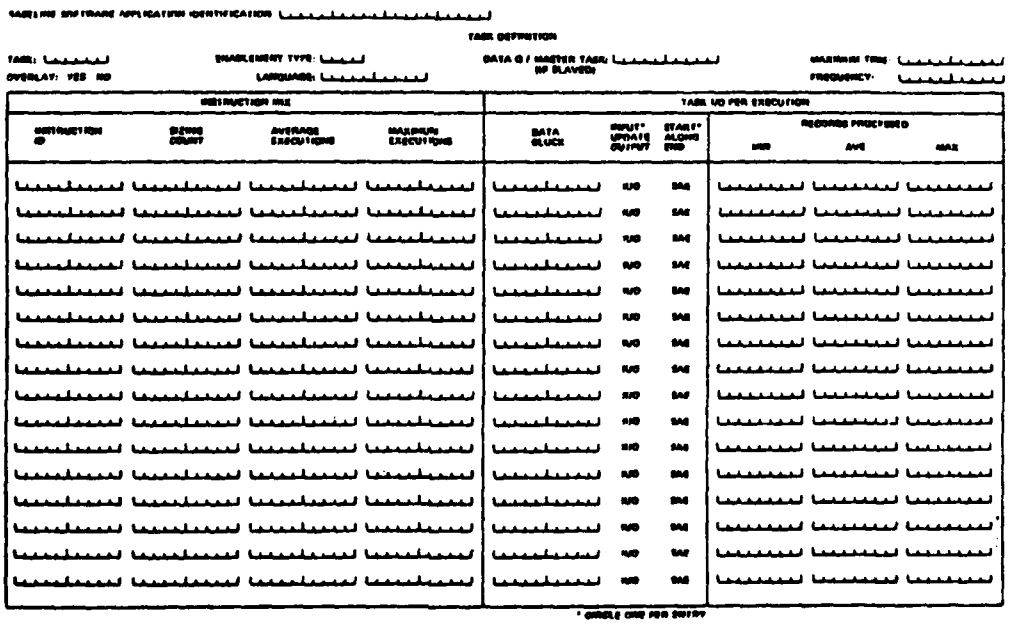


BASELINE SOFTWARE APPLICATION IDENTIFICATION

DATA BLOCK DEFINITIONS

ID	LEVEL	DISCIPLINE	SYSTEM INTERFACE DEVICE	MAXIMUM RECORDS	NUMBER BITS/ BYTE	NUMBER BYTES/ WORD	RECORD SIZE IN WORDS MINIMUM	AVERAGE	MAXIMUM

Figure 21. **Data block definition.**



BASELINE SOFTWARE APPLICATION IDENTIFICATION

TASK DEFINITION

TASK: DISABLEMENT TYPE: DATA G / MASTER TASK/ (IF SLAVED) MAXIMUM TIME:

OVERLAY: YES NO LANGUAGE: FREQUENCY:

RESTRUCTION RULE				TASK I/O PER EXECUTION					
INSTRUCTION ID	SKIPPED COUNT	AVERAGE EXECUTIONS	MAXIMUM EXECUTIONS	DATA BLOCK	INPUT* UPDATE OUTPUT	STABLE* ALONG EDGE	RECORDS PROCESSED MIN	AVE	MAX
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			
					I/O	SAE			

* CIRCLE ONE FOR ENTRY

Figure 22. **Task definitions.**

EVALUATION RUN IDENTIFICATION

PARTITIONING ASSIGNMENT

	COMPONENT ASSIGNMENT D - DATA T - TASK	APPLICATION COMPONENT IDENTIFIER	CANDIDATE CONFIGURATION COMPONENT	
			IDENTIFIER	VALUE IF APP.

Figure 23. **Allocation inputs for allocation and memory data/instruction block storage.**

4.3.3 *Baseline Load.* The baseline load is the means of representing the computational subsystem external environment. The basic underlying organization is a training exercise timeline of external I/O activities and the identification of the required computational paths to be simulated and evaluated. The following groups of Table B-4 Appendix B were designed to facilitate load from the steady state mix to more intricate I/O relationships of the real-time environment.

1. *Group 1 — Baseline Load Identification* provides a unique label for all the simulation run results plus the evaluation timeline duration in terms of a start and stop time.

2. *Group 2 — Statistical Model Initialization* permits definition of the statistics to be collected, frequency of collection, and random number generator seed values.

3. *Group 3 — Master Sequences* permit a task loading to be defined as groups of data arrivals, cycles and/or threads which are scheduled to begin and/or end at some point in the simulated time line.

4. *Group 4 — Data Arrivals* permit data specific paths to be modelled and evaluated for potential queuing problems such as processing delays and lost records.

5. *Group 5 — Data Processing Cycle* parameters describe a group of task or task threads which must be executed at a given time enablement frequency.

6. *Group 6 — Data Processing Threads* permit a sequence of dependent tasks (which generally perform a given I/O transformation) to be defined which may be triggered via a master sequence or a cycle within a master sequence.

Additional design details for a given implementation are necessary before a complete set of recommended input forms can be drawn up. This is especially true for statistical collection options. As a

simple example, the basic size of the candidate configurations being evaluated has a direct influence on the sizing for statistical data collection. A small (three processor, one memory, fifty task) system versus a larger (twenty processor, ten memory, three hundred task, four hundred block) system influences the data structure and selective options necessary to provide meaningful, organized statistics collection. For small systems (generally a subsystem analysis) it may be most economical to collect and reduce statistics as part of the functional simulation. For larger systems a statistics log may be output and then reduced in subsequent postprocessing report generators to extract desired data.

The time tagged data is more straight forward. The master sequence may be represented utilizing the form in Figure 24. This is complimented with appropriate Data Arrival forms (Figure 25), Data Processing Cycle Forms (Figure 26), and Data Processing Threads (Figure 27).

DYNAMIC LOAD IDENTIFIER

MASTER SEQUENCES

SEQ	START	STOP	TYPE (CIRCLE ONE)	IDENTIFIER
1			D C T	
2			D C T	
3			D C T	
			D C T	
5			D C T	
6			D C T	
7			D C T	
8			D C T	
9			D C T	
10			D C T	

Figure 24. **Master sequence timeline.**

DYNAMIC LOAD IDENTIFIER

DATA ARRIVALS

IDENTIFIER	BLOCK ID	TYPE	DISTRIBUTION PARAMETERS	

Figure 25. **Data arrival parameters.**

PROCESSING CYCLE

CYCLE IDENTIFIER	ITERATION RATE (PPS)	TASK/THREAD LIST		

Figure 26. **Processing cycle definition.**

PROCESSING THREAD

IDENTIFIER THREAD	LEVEL	PREDECESSOR LEVEL	TASK LIST

Figure 27. **Processing thread definition.**

4.3.4 *Technology Data Base.* To assure uniform modelling and evaluation of various alternate candidate configurations, an up-to-date technology data base is essential. Recommended technology attributes are grouped as follows:

1. *Group 1 — Technology File Identifier* permits identification and verification of specific file being used for technology parameters. Thus, multiple files could be established by various user organizations.

2. *Group 2 — Master Technology Categories* provides for identification of both general purpose and special purpose devices by categories such as processors, communication link, cockpit controls, etc. A tentative list has been given in Appendix B, Table B-5. For each category, the number of entries are currently defined, and a master identifier for each category is maintained.

3. Group 3 through (TCNC+2) represent the specific parameters in each of the TCNC respective technology categories. These parameters have a direct relationship with the model capacity and fidelity. They are subject to refinement as details evolve for any specific model implementation.

Table B-5 represents minimal features which may be modularly expanded as additional model technology capabilities are defined. The generic handling of the other inputs (defined above via appropriate identifiers) permits the user to set up a variety of configurations without having to script all the detailed attribute data once a given device has been defined in the data base.

A set of recommended input forms to describe processors is illustrated in Figures 28 and 29. Memory device parameters are grouped in Figure 30. Communication link definition form is presented in Figure 31.

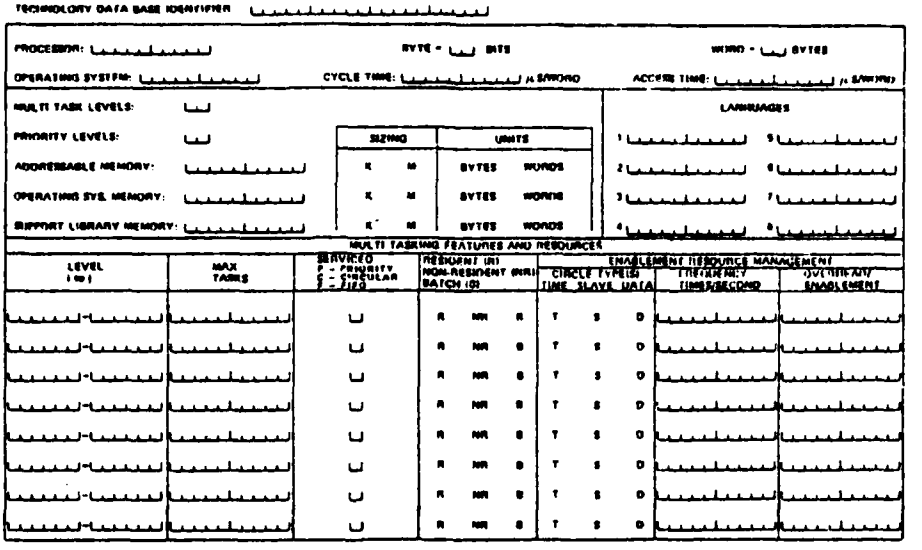


Figure 28. **Processor description.**

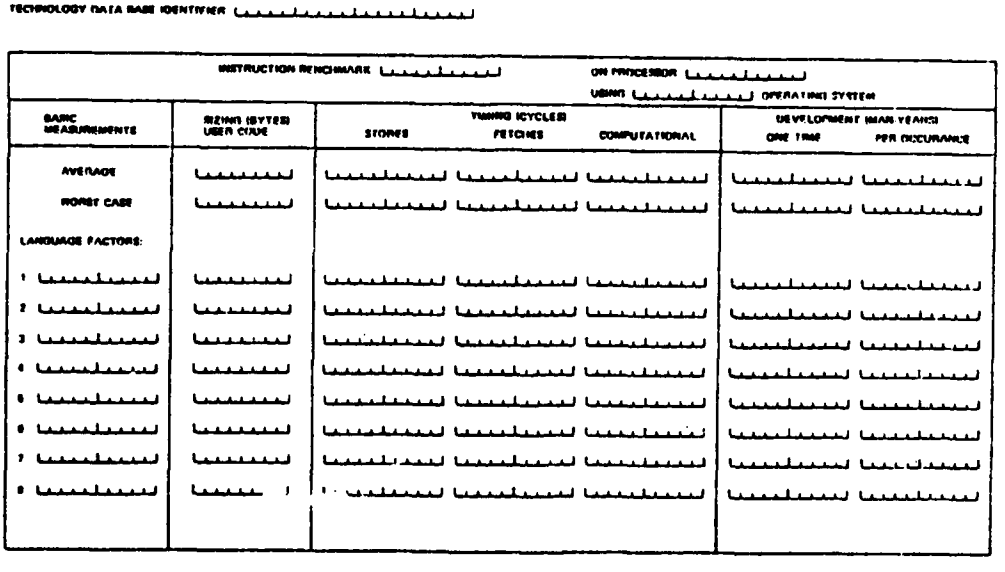


Figure 29. **Processor instruction set timing, sizing, and development inputs.**

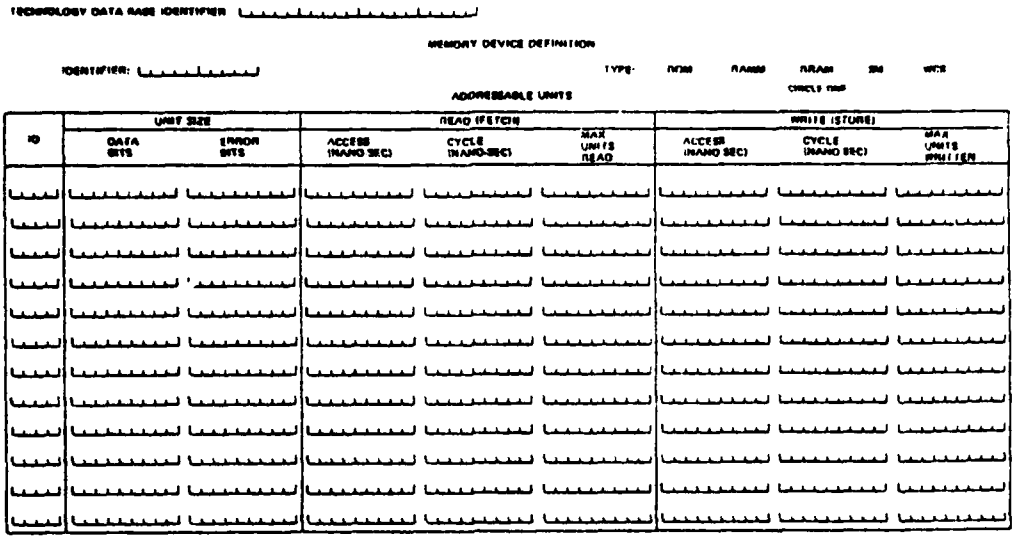


Figure 30. **Memory device definition sizing and timing features.**

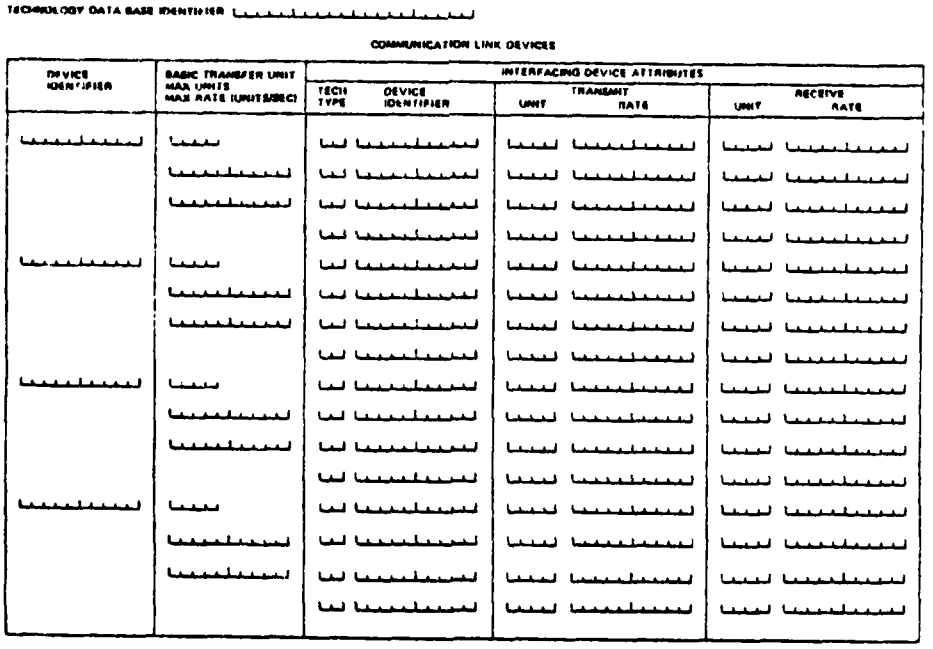


Figure 31. **Communication link interface features.**

4.3.5 *Evaluation Options.* To facilitate computational configuration design evaluation analysis, this area of user inputs provides the basic evaluation user interface once the files for the areas described above have been established. The three groups which comprise this input area (Table B-6) permit proper run quality and configuration controls, modelled load or design parameter changes, and required limits for spare and/or task related execution times. Group definitions include:

1. *Group 1 — Evaluation Run Identification* provides the user a means of identifying and verifying that the proper files for interface requirements, technology, candidate design, and baseline load databases are being used for a specific evaluation run.

2. *Group 2 — Evaluation Options* facilitate a streamlined setup for parametric study runs to aid in determining the performance tradeoffs and potential growth limits of the design under evaluation. At the completion of a detail design there are still unknowns in timing and sizing estimates. These options permit cycle iteration frequencies, data arrivals, task timing, and/or data volumes to be increased or decreased for a given evaluation run to help obtain a handle on the candidate design sensitivity to various loads.

3. *Group 3 — Required Measures* provide a means for denoting processor, memory and communication channel performance utilization with respect to growth requirements as was illustrated in the output parameters via flagged report items. Also included in this group are task cycles and/or threads for which throughput measures are to be collected with respect to evaluator specified times.

The specific evaluation environment must be addressed in order for a set of meaningful, useable forms can be established. Therefore, input forms for this area have been left as an implementation design decision.

4.4 Processes

Several existing performance measurement simulations were surveyed to initiate a list of desired design features and models to be incorporated as is or as model candidates to be modified or expanded in the design of the computer performance predictor simulation model. Simulations which were studied include:

1. SAINT — Systems Analysis of Integrated Network of Tasks from the Aerospace Medical Research Laboratory, AFSC WPAFB

2. DPSIM — Data Processing System Simulator from TBE

3. CRAM — Critical Resource Analysis Model, including data processing and C_3 models, from TBE.

All of these simulations are FORTRAN based. The first two are derivatives from the GASP simulation language. The third (CRAM) is based on TBE's simulation language, MODELER. Both GASP and MODELER facilitate discrete event, continuous event, and dynamic file management features described in the following design process descriptions. All of these simulators incorporate generic models and flexible user interfaces to support a variety of analysis modelling needs, statistical collection, and quick look report formatting.

Each has specific models which are the results of specific simulation modelling applications. SAINT users have expanded upon the man machine interface models. DPSIM emphasizes modelling of multiple processor resource management rules for multitasking real-time application environments under various open loop and closed loop load scenarios. CRAM addresses conceptual man machine environments with specific models for constructing and analyzing alternative communication networks and data processing network configurations.

It was determined that no one simulator has all the desired features or models desired for the multiple processor performance predictor simulator. However, any one of these could be adapted and expanded to provide a flight training simulator multiple processor design performance prediction evaluation tool. Our design has remained independent of an existing simulator to permit the delineation of specific model processes necessary to meet the design goals.

We now describe the processes necessary to transform the inputs of Section 4.3 into the output of Section 4.2. These are grouped into three major steps of the computational design performance predictor model (as defined in Figure 32): namely, initialization, computational subsystem simulation, and summary report generation. The simulation step is the heart of the processing and is described in more detail. As will be evident in Section 4.4.1, the required input initialization and output report summary processes are a function of respective model input and output parameters chosen for a given implementation evaluation environment.

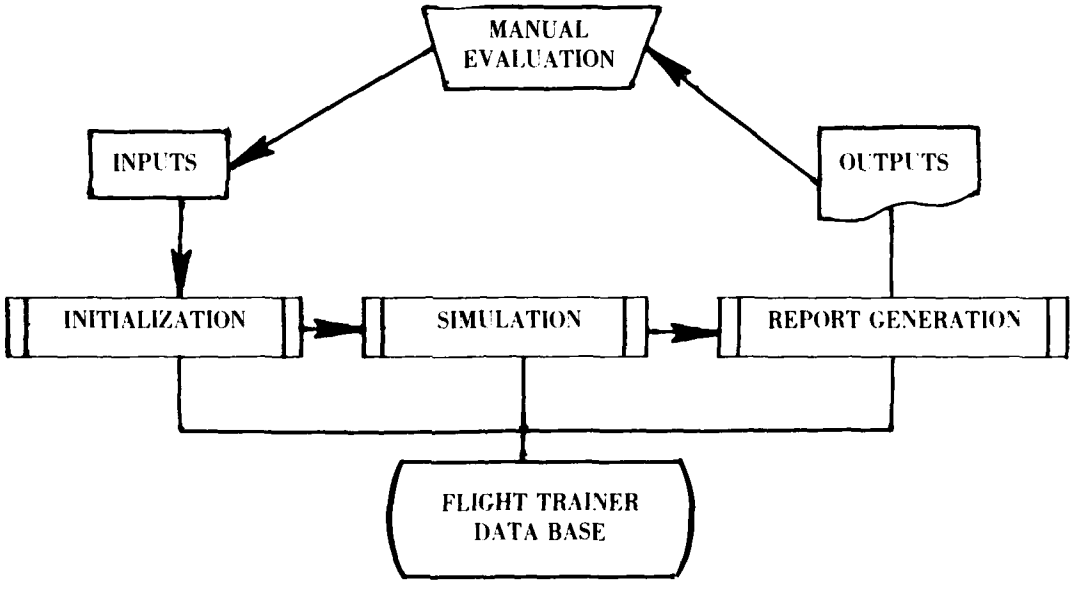


Figure 32. **Three major model processes are initialization, simulation, and report generation.**

Table 8 summarizes the basic software modules which comprise the Computational Performance Predictor Simulation Model (CPPS). The modules have been categorized into the major functional areas of initialization, simulation control, application model events, candidate hardware events, statistical collection and end simulation. The design considerations for each of these functional areas are now presented. The initialization and end simulation functional areas have been consolidated into data base I/O features.

Table 8. **Computational Performance Predictor Simulator Model Modules**

Major Function	Module		Description
	PPD	Mnemonic	
Initialization	1000	CPPS1	Input and initialization
	1100	CP1EO	Process evaluation options
	1200	CP1IR	Process computational interface requirements
	1300	CP1DA	Process candidate design allocation
	1400	CP1BL	Process baseling load inputs
Simulation Control	2000	CPPS2	Discrete event simulation
	2100	CP2DM	Event trace if debug monitor on
	2200	CP2EV	Perform next event
Application Model Events	3100	PULGEN	Application load pulse generator
	3300	ARRVL	Application pulse arrival into a given queue
	3500	LEAVQ	Application pulse removed from given queue for servicing
	3700	PEXIT	Application pulse service completed
Candidate Hardware Events	4000	PROCCK	Processor event state change rules
	4100	PROCGN	Processor generically modeled
	4500	PROCEX	Processor explicitly modeled
	5000	MEMCK	Memory event state change rules
	5500	MEMEX	Memory explicitly modelled
	6000	COMCK	Communication event state change rules
	6100	COMGN	Communication generically modeled
	6500	COMEX	Communication explicitly modeled
	7000	EXTDCK	External device event state change rules
	7100	EXTDGN	External device generically modeled
	7500	EXTDEX	External device explicitly modeled
Statistical Collection Summary	8000	STATCL	Statistical collection update
	8100	STATPS	Periodic summary
	8200	STATGS	Global statistical summary
	8300	STATRS	Statistica Reset
End of Simulation Processing	9000	ENDSM	End Simulation

4.4.1 *Data Base I/O.* Prior to functional simulation the initialization step involves both manual and automated process (preferably interactive) steps to collect, format, sort, and merge appropriate data inputs. Detailed design of these steps requires that the specific evaluation environment objectives and existing data collection formats be studied. From this study, automated processing for interfacing, converting, and/or additional collecting of inputs can be designed and implemented for a given evaluation organization. The input parameters and forms described in Section 4.3 provide a starting point as to the hierarchical categories of a data base which address the basic generic multiple processor configuration as applied to a given real-time flight training application environment.

The end simulation report generator design features are also greatly influenced by the data base structure, as well as, the evaluation environment. The major advantage of an independent report generator (over built-in standard set of simulation run reports) is that it can be used at any time after the run to generate selective reports as needed. For example, if a memory component is flagged as a bottleneck, the memory processor communications report for the suspect memory can be requested for further details to help identify the storage/retrieval blocks in demand. By utilizing an interactive multi terminal report generator process, several evaluators could be addressing separate aspects of the alternative candidate design configurations under evaluation.

This centralized design repository concept for I/O, if properly implemented, facilitates configuration management and, in turn, the resulting analysis integrity associated with comparative and related tradeoffs is of higher quality. Care should be taken in polling the potential users of the system prior to selection and detail design of the automated repository tools to be implemented. Commercially available data base systems have many features which should not be overlooked. As a minimum the following features should be addressed:

1. Collection sources and formats
2. Volume and frequency of data I/O
3. Security provisions
4. Configuration management controls
5. Backup recovery from system failure
6. Data accessibility via multiple users and programs
7. Hierarchical and relational storage/retrieval
8. Report generation features.

The current functional simulator languages do not provide these features. For this reason, most of them operate in a batch mode or a customed interactive environment which incorporates voluminous pre-formatted files. The automated data base area is one recommended for further study.

Figure 33 illustrates the segmented data base structures which have been designed to interface with both CPPS (block 10) and the Partitioning Algorithm for Software Systems (block 9) under given evaluator (block 3) run requests. The left hand side represents the flight simulation requirements, baseline load, technology and candidate design SW/HW definitions. The right hand boxes represent outputs which are available to the evaluator during the iterative process of computational partitioning and performance prediction evaluation. This study has concentrated on the performance prediction tool. Reference 7 relates design details applicable to the partitioning evaluation tool. A good data base structure will greatly facilitate a structured, well organized set of evaluation tools.

4.4.2 *Simulation Control.* Discrete event simulations are controlled and driven by discrete time tagged events. An event calendar permits a centralized mechanism for posting of events and determining the next event to be processed. A set of event handling rules should include the ability to:

1. Support parallel events (more than one activity model for a particular time line instance).
2. Recognize and avoid duplicate postings of a redundant type (identical) event for the same time.
3. Permit a priority scheme for event execution selection for events which occur at the same time.
4. Permit an event which is currently posted to be rescheduled or deleted.
5. Utilize a dynamic memory management scheme for maintenance of the event calendar and user dynamic files.

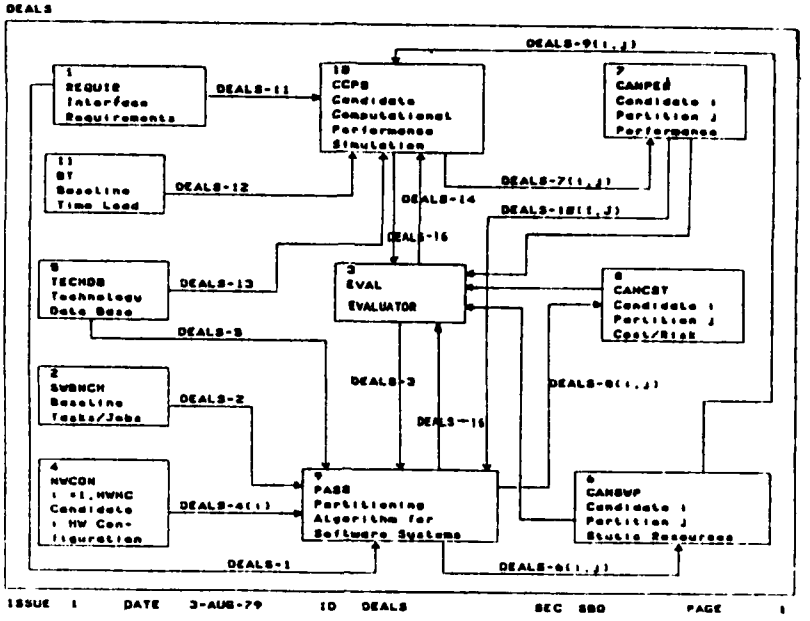


Figure 33. **Relationship of partitioning and simulation models.**

6. Maintain at least four event attributes (one of which keys the specific event) which may be either real and/or fixed dependent upon event definition.

7. Accommodate space to handle at least 1000 events at any point in the simulation.

8. Permit interface to user supplied event code to model the event.

9. Access to complete source code listing of any and all event calendar simulation code to be incorporated in actual implementation.

10. Provide debug mechanisms for event trace printouts/displays as a function of user supplied simulated time intervals of interest.

It should be noted that both GASP and MODLER support most of these features when the user supplied model code adheres to certain coding conventions inherent to the respective host simulation language.

The models to carry out the computations for performance prediction of a candidate flight training simulator computational subsystem design are illustrated in Figure 34. These models are under the control of a discrete event simulation event calendar as defined in this subsection. The application models include the external I/O load/response events, data block models, and task models. The hardware configuration models include the memory access management events; processor operating system and task service events; and the communication rules models and events. Statistical collection models and events interface with application and configuration models to collect queuing and timing statistics.

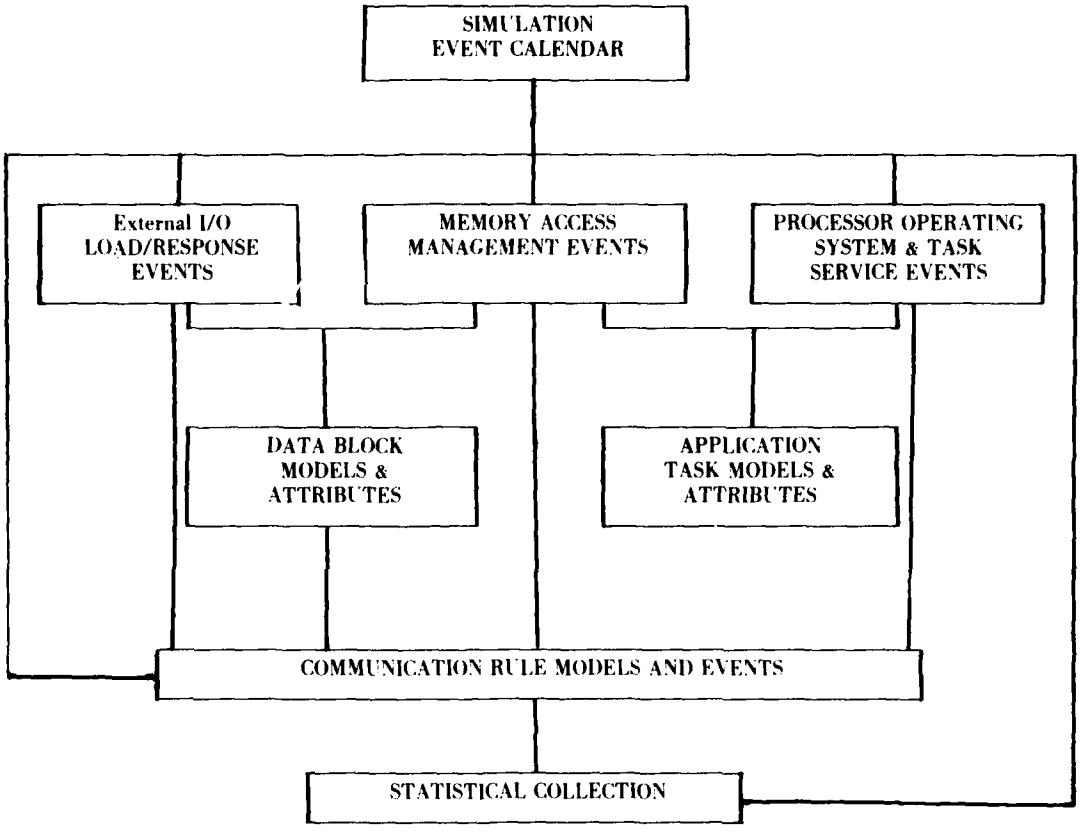


Figure 34. **Discrete events in the model include external I/O load/response, memory access management, processor operating system and task servicing, and communication rules which are driven by candidate design attributes for the load, components, data, task and communication flow.**

The basic event scenario for the generic multiple processor is one of data arrival and/or time triggered task entries which require service by communication and processor devices. The respective devices are then busy for a given period of time. At the end of this time period a response and/or other events are generated. The device becomes idle if no other entries have been placed in its queue waiting to be serviced. Each of the major events is further described under its specific functional relationship with the performance predictor simulator.

4.4.3 *Real-Time Computational Application Models and Events.* The real-time computational subsystem is characterized via a baseline load which is serviced via data queuing and predefined processing tasks. The cyclic nature of the flight training simulator computational tasks facilitates load initiation and task tracking events. The inputs described in Section 4.3.2 and 4.3.3 provide the basic computational task design and load parameters from which task flow may be simulated.

The basic load may be likened to a pulse generator. A given load is referenced as a master activity sequence such as cockpit A computational flow during takeoff. Another master sequence might represent enroute flight configuration computational mix and yet another for air-to-air combat engagement. The start and stop times provide a means to vary the training mix activities with respect to time for each training station to be serviced by the candidate computational system design. The input load/flow parameters provide the following discrete events:

1. Master activity start/stop.
2. Cycle/thread initiate, data track, complete, repeat if cyclic.
3. External data queue/buffer record generation input/output.

The major emphasis in determining performance is on data flow and throughput. Figure 35 illustrates the discrete application events with respect to data/task flow. Cycle and thread definitions are designed to act as a feedback mechanism for pulse arrivals.

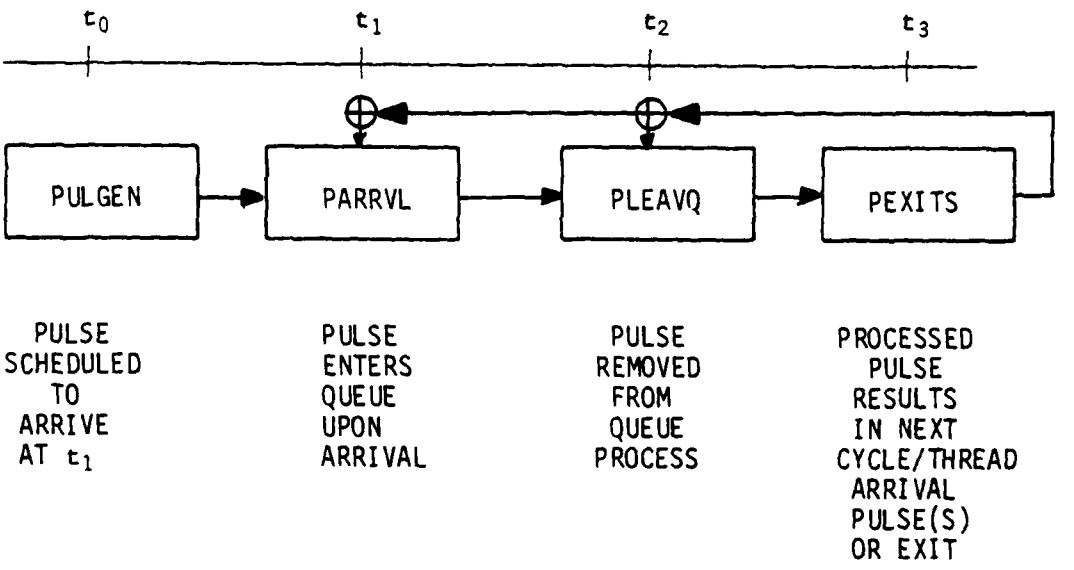


Figure 35. **Data/Task enablement pulse flow through system states of generation, arrival, leave queue, and exit.**

The word task queue is utilized to distinguish that the type of data block being modeled is related to task load enablement. It may represent a physical data queue block in memory or a time triggered processor task enablement mechanism. Any and all data blocks to be communicated among the application tasks and/or from external devices are defined via the user candidate design Group 4 inputs (Section 4.3.2). Simulation models of data flow for each of the data block disciplines (FIFO, LIFO, Circular buffer, sequential, random etc.) permits timing and sizing computations to be made taking the type of I/O and communication device into consideration. These are support models which are utilized by event logic steps when modelling data flow. Random number generated distribution models for probabilistic data paths are also incorporated to provide parametric timing sensitivity studies.

Specific task resource requirements are handled via model task allocation inputs of the candidate design Group 5 (Section 4.3.2). These attributes are used to define the scheduling levels and schedule states applicable to each processor and to trace the simulation from the application viewpoint to collect appropriate statistics for tasks, cycles, and threads. These attributes and models are utilized as needed by the major configuration events. For example, if Task T is to execute on processor P, the time for its completion is computed as a function of task T's description and processor P's technology data base statistics. Communication models are referenced to obtain I/O execution times for task communication. We now describe the candidate configuration events.

4.4.4 *Candidate Configuration Hardware Models and Events.* The candidate configuration models include processors, memories, communication, and external device models. As denoted in Table 8 design provisions have been made for device state rules and state transition with both generic and/or user supplied models of a given device. Each of the four generic devices is briefly described.

The processor event models are designed to facilitate a variety of architectures and operating system features to include the following:

1. Priority and/or circular scheduling of time enabled, data enabled, and/or slaved tasks on a given real-time schedule loop (such as frame, sub-cycle, cycle, batch etc.) is modelled.

2. Multitasking of at least four scheduling level task loops incorporating I/O interleaving and scheduling level discipline rules is modelled.

3. Task I/O interrupts are interfaced with communication events to determine service delay time for resumption of task as a function of the required processor, data block, and device communication attribute.

4. Statistics based on processor utilization events are to be collected for use in reports as described in 4.2.

Design efforts placed emphasis upon generic processor operating system features. Simulation mechanisms were studied for modelling of the various discrete processor states to include task enablement, task execution, and task interrupt handling procedures. As a result of this design analysis, additional processor technology and baseline application input parameters were identified. The technology parameters (Table B-5 Appendix B) identified relate directly to the level of multitasking and means for introducing and invoking tasks in a multiple processor environment. These parameters relate to operating system selection and customed application real-time executive controls.

The simulation performance models studied tend to lump application execution resources and operating system overhead together. This technique is appropriate for a distributed executive when the application tasks utilize the same processor as the decision making process. However, in instances where a master/slave scheme is utilized the application tasks are not run on the same processor as the executive controlling processor. In this latter instance there is a need to decouple the task enablement decision making processes from the application task. The multiple processor model provides a means to define both distributed executives and master/slave executive processing disciplines. It basically requires the executive task(s) be included as separate task definitions. This is most important in the master/slave configurations.

Design emphasis was also placed on algorithms for modelling multiple-processor relationships with shared memory. The increasing utilization of multiport memories, memory mapping, and communication bases has facilitated additional memory capacities and interprocessor communication alternatives for user applications.

These features simplify user application code interfaces but carry with them additional operating system and communication resource management overhead. One critical aspect is the detection, prevention, and/or resolution of deadlock situations. This is the case when two processes are waiting for access to a resource currently dedicated to the other process. In this case neither can proceed without an arbitrator which can suspend one of the processes and properly redo resource assignments with appropriate data integrity controls for the given application. The processor models designed for the performance predictor algorithm includes a means for detecting deadlock situations. This is basically done by the memory model status and wait time exceeded indicators. The model could also incorporate various analytic algorithms for bottleneck detection to support more detailed analysis of shared resource benefits and shortcomings.

There are basically two memory management events: access request made and access request complete. The access request utilizes current memory state and data block read/write access request information to compute delay before data access will be completed. A request complete event is posted for the time in which the data will be made available to the requesting processor or peripheral. These events will be applicable to each memory device in the design configuration and also serve to collect memory utilization statistics.

To properly relate memory/processor external device/processor, processor/processor, and external device/memory communications, communication models and events are necessary to close the loop. These models permit the rules of the candidate computational subsystem to be modeled and evaluated. This particular area has been selected for further analysis as to ascertaining the critical performance features which should be modeled and measured for both the full operational design and the impact of devices which may be temporarily out or off-line. As a minimum the simulation model should account and readjust multiple communication requests for the same transmission link to the predetermined sequential order via the configuration rules. This is to include check for target device availability as well as communication component utilization statistics of wait, busy, and idle states.

4.4.5 *Statistical Distribution and Collection Models and* Events. A major aspect of discrete event simulation is the means for setting up, updating, and reporting statistical measures for a variety of parameters. Statistical collection is greatly simplified if a set of standard generalized routines are available for basic parametric statistics, time utilization statistics, histogram statistics, and special distribution computations. Basic statistics include number of observations made, mean, standard deviation, maximum observed, and minimum observed values for given user specified parameters. Time utilization is based on zero-one status summed over a given time interval; thus, it represents the length of time spent on a given state such as busy (one) versus idle (zero) time. Histograms keep track of the observed value distribution in terms of a finite number of intervals which generally denote a unit of time for which the observation is made. Special distributions (such as uniform, normal, exponential, gamma, poisson, etc.) permit parameterization of modeling decision steps as well as special output report computations. Both GASP and MODELER provide for statistical data collection and reduction.

4.4.6 *Final Design Notes.* The generic multiple processor model, as defined by this study, must contend with an ever changing environment of alternatives. The complexity of the communication modeling alternatives was recognized early in the contract as requiring more study than alloted time and resources could permit. As a result the design details and manual demonstration problem as presented in this report had to be kept at a fairly high overview level. A more rigorous design and demonstration are possible given more analysis time and resources.

5.1 Findings

The ultimate test of a design is to build a prototype to see if it indeed works. This is an expensive step which may encounter design flaws in the process of debugging the design implementation causing time delays due to design alterations in both hardware and software. This study has pursued techniques for design evaluation analysis prior to its prototype implementation. In particular the candidate configuration task activity time line with respect to resource management, communication, and operational rules has been stressed.

The flight training environment is one of dynamic changing loads which reflect current training job mix in progress. Certain job mix capabilities are naturally more stressing than others. For advanced training configurations (with multiple crew training stations and instructor stations) enumeration of all possible job mix states may be an enormous if not impossible undertaking in itself. However, identification of the tasks required to support a given required training capability is an essential feature which must be traceable to and accounted for by any viable candidate design in terms of a representative demonstration which provides traceability of all systems inputs through respective task processes necessary to obtain the resultant system responses.

In the multiple processor environment manual demonstrations of the design for particular capabilities is time consuming, error prone, and often too bulky to look at all the intricate details. Two automated techniques may be employed, namely, algorithm emulation and system simulation. Emulation requires that a machine level model be available to represent the target processor and the environment in which it is to operate including the actual software object code and realistic sample input data to be processed. This is a good technique for benchmarking of algorithms for hardware which is on the design drawing board but not yet available. However, it is not suitable for evaluation of the total system design and internal system configuration interactions from a multiple processor system design evaluation point of view. A system simulation of the design components for given baseline loads is the only viable alternative for evaluating the candidate design in terms of timing and data flow for a proposed operational training simulator system prior to prototype availability.

Many types of system simulations exist, but in most cases they address specific rather than generic models making them good tools for a given project but very inflexible and unappropriate for use with other projects. One reason is the lack of standard rigorous design notation. A second reason is the use of customized assumptions which are built into the modeling terminology definitions and output interpretation for a given project. In many cases these design model simulations are products of the designer developing the actual system and the benefits of independent evaluation are not obtained.

The resultant evaluation model design derived by this study is a generic multiple processor candidate configuration simulation which is independent of any specific system with regard to processor, memory, and communication models. It is driven and based upon data which reflects a complete design for a given candidate under evaluation including the software task relationships for given baseline evaluation loads. The model (described in Section 4) includes automated process steps for data base input/output management necessary to support detailed and quick look evaluation analysis.

5.2 Recommendations

Two areas requiring further study and analysis were identified. These are:

1. Communication technology and design reliability alternatives.

2. Automated flight training simulator candidate design data base repository.

These two areas relate directly to the level of simulator fidelity and feasibility of its use as a standard design evaluation tool.

To carry out these studies, it is recommended that a specific Air Force Agency (which is responsible for flight training computational design evaluations) sponsor a prototype implementation of the functional simulation model. Details of the communication and data base tasks would constitute the first phase of the recommended effort to delineate the complete detail design for the sponsor agency. The second phase, upon approval of Phase I, would be used to automate, test and verify the design.

6. REFERENCES

1. **Rivers, H.A.** *Final report simulator comparative evaluation.* Tactical Air Command USAF Tactical Air Warfare Center, AD-B023450, November 1977.

2. **Sigmund, F.A.** The efficiency of FORTRAN in simulation computers. *10th NTEC/Industry Conference*, 15-17 November 1977.

3. **Pirrelo, C.J., Hardin, R.D., Capellupo, J.P., & Harrison, W.D.** *An inventory of aeronautical ground research facilities.* Volume IV Engineering Flight Simulator Facilities, NASA CR-1877, November 1971.

4. **Babel, P.S.** *Use of floating point in programming realtime digital flight simulators.* Ohio State University (Thesis), 1969.

5. **Tucker, D.E.** Downtime wastes the resources. *10th NTEC/Industry Conference*, 15-17 November 1977.

6. **Harris, E.** *Simulator system engineering the computer allocation process.* University of Dayton, March 11-14, 1980.

7. **Clymer, S.J.** *Software partitioning schemes for advanced simulation computer systems.* AFHRL-TR-80-42. Williams AFB, AZ: Operations Training Division, Air Force Human Resources Laboratory, 1980.

APPENDIX A: FUNCTIONAL PERFORMANCE SIMULATOR
OUTPUT DEFINITIONS

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	CONFIGURATION COMPONENT PERFORMANCE OUTPUTS			
	Basic utilization for each component			
	. Component Category	POCC		Technology category
	. Component Identifier	POCID		Candidate system identifier
	. Busy	POCB	F18.6	Seconds
	. Idle	POCI	F18.6	Seconds
	. Flag	POCF	2 characters	
			=blanks	spare requirement met
			='**'	spare requirement not met
	Processor utilization for each processor p of the candidate configuration			

ISSUE 1 DATE 3-AUG-79 ID DEALS SEC IOPT-7 PAGE 1

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Application utilization for each task t executed on processor p attribute i statistic j	POPTU (p,t,i,j)		j=1 implies minimum j=2 implies maximum j=3 implies mean j=4 implies standard deviation
	-Computational	POPTU (p,t,1,*)	F10.6	Seconds
	-I/O	POPTU (p,t,2,*)	F10.6	Seconds
	-Suspended(by another task)	POPTU (p,t,3,*)	F10.6	Seconds
	-Total	POPTU (p,t,4,*)	F10.6	Seconds
	.For each task t on processor p a total time requirement flag	POPTF (p,t)	3 Characters	
			=blanks	timing requirement met
			='*'	*worst case* violates timing
			='**'	average case violates timing
			='***'	minimum case violates timing

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Total for all application execution on processor p	POPTA(p)	F18.6	Seconds
	.Total for resource management/operating system overhead on processor p	POPRO(p)	F18.6	Seconds
	.Idle	POPID(p)	F18.6	Seconds
	Memory utilization for each memory m of the candidate configuration			
	.Read (a=1)/Write (a=2) services of memory m for processor p attribute i=1,3	PORWP (m,a,p,i)		
	-Number transfers	PORWP (m,a,p,1)	F7.,GE.0	
	-Minimum data transfer size	PORWP (m,a,p,2)	F7.,GE.0	User selected unit bits/bytes/words
	-Maximum data transfer size	PORWP (m,a,p,3)	F7.,GE.0	User selected unit bits/bytes/words

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Mean data transfer size	PORWP (m,a,p,4)	F7.,GE.0	User selected unit bits/bytes/words
	-Standard deviation for data transfer size	PORWP (m,a,p,5)	F7.,GE.0	User selected unit bits/bytes/words
	-Minimum time for transfer	PORWP (m,a,p,6)	F10.4,,GE.0	Micro Seconds
	-Maximum time for transfer	PORWP (m,a,p,7)	F10.4,,GE.0	Micro Seconds
	-Mean time for transfer	PORWP (m,a,p,8)	F10.4,,GE.0	Micro Seconds
	-Standard deviation time for transfer	PORWP (m,a,p,9)	F10.4,,GE.0	Micro Seconds
	-Number access requests	PORWP (m,a,p,10)	F7.,GE.0	
	-Minimum access wait time	PORWP (m,a,p,11)	F10.4,,GE.0	Micro seconds

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Maximum access wait time	PORWP (m,a,p,12)	F15.4,.GE.0	Micro Seconds
	-Mean access wait time	PORWP (m,a,p,13)	F15.4,.GE.0	Micro Seconds
	-Standard deviation access wait time	PORWP (m,a,p,14)	F15.4,.GE.0	Micro Seconds
	-Total wait time	PORWP (m,a,p,15)	F15.0,.GE.0	Micro Seconds
	-Total transfer time	PORWP (m,a,p,16)	F15.0,.GE.0	Micro seconds
	.Data block b read (a=1) / write (a=2) service of memory m (for any and all processors) attribute i=1 to 19	PORWB (m,a,b,i)		
	-Attribute i=1 to 16 same as for processor PORWP (m,a,*,i)			see previous parameter definition

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Data block length for variable blocks only (i.e. queues,stacks, lists,etc)			
	.Miminum	PORWB (m,a,b,17)	F18.1,.GE.0	Records
	.Maximum	PORWB (m,a,b,18)	F18.1,.GE.0	Records
	.Mean	PORWB (m,a,b,19)	F18.1,.GE.0	Records
	.Standard Deviation	PORWB (m,a,b,20)	F18.2,.GE.0	Records
	.Records lost to overflow	PORWB (m,a,b,21)	F18.0,.GE.0	Records
2	CYCLE/THREAD PERFORMANCE OUTPUTS Basic parameters for each cycle or thread ! throughput			

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Number of occurances in evaluation time period	POCTO(I)	F6.0,.GE.0	
	.End-to-end timing statistics J=1 to 4	POCTS(I,J)		
	-Minimum	POCTS(I,1)	F8.6,.GE.0	Seconds
	-Maximum	POCTS(I,2)	F8.6,.GE.0	Seconds
	-Mean	POCTS(I,3)	F8.6,.GE.0	Seconds
	-Standard Deviation	POCTS(I,4)	F8.6,.GE.0	Seconds
	.Flag	POCTF(I)	3 characters	
			=blanks	timing requirements met
			='* '	worst case timing violates requirements
			='** '	average timing violates requirements
			='***'	minimum timing violates requirements

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Any of selected group 1 parameters collected for given cycle or non-overlapping thread occurance			See Group 1 and related input variable of IOPT-3 Group 4.

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
3	CRITICAL PATH IDENTI-FICATION			
	Number critical paths	PPNCP		
	For each critical path c			
	.Number of tasks [PONCP]	PPNTA(c)		
	.For each task [PONCP,PPNTA(c)]			
	-Task Identifier	PPTID (c,t)	6 character mnemonic	Must match task identifier in candidate design (IOPT-9 group 5)
	-Critical Resource Code	PPCRC (c,t)	4 characters	
			='I/O'	significant I/O wait time
			='PROC'	compute bound
	-If I/O wait time is the critical component identify data block	PPCRB (c,t)	6 character mnemonic	must match block identifier in candidate design (IOPT-9 group 4)

APPENDIX B: FUNCTIONAL PERFORMANCE SIMULATOR INPUT DEFINITIONS

Table B-1. Computational Interface Requirements

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	SYSTEMS REQUIREMENTS FILE ID	RSRID	20 Characters	Used to uniquely identify system being specified
2	SYSTEM INTERFACE REQUIRED DEVICE			
	For each component			
	.Identifier	RICID	10 characters	Components Identifier to map into candidate configuration component (IOPT-4)
	.Technology type	RICTT	2 character code	Must match an entry in the technology category codes as defined in the technology data base IOPT-5 group 2
	.Specific Device Identification	RICTD	18 characters	Must match to device in technology data base IOPT-5 group 2 based upon category type

ISSUE 1 DATE 27-NOV-79 ID DEALS SEC IOPT-1 PAGE 1

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Required options for Interfaces where k=1 to number of options for Technology type RICTT(i)	RICOP (k)	Dependent upon RIC	Options must be specified to match format in technology data base IOPT-5, group.
3	SYSTEM DATA BLOCK/ BUFFER			
	For each data block			
	.Block Identifier	RSDBI		
	.System device identifier to which assigned	RSDBD		
	.Discipline	RSDAT (1)	4 character code	Same as DEALS-IOPT5 Group 2 master data discipline codes
	.Maximum Records	RSDAT (2)	Positive Integer	
	.Record Length	RSDAT (3)	Positive Integer	BITS
	-Bits/Character			
	-Characters/Word	RSDAT (4)	Positive Integer	Characters

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Minimum Words	RSDAT (5)	Positive Integer	words
	.Maximum Words	RSDAT (6)	Positive Integer	words
	.Average Words	RSDAT (7)	Positive Integer	words
4	SYSTEM FUNCTION REQUIREMENTS			
	For each function			
	.Function Identifier	RSFID	10 characters	
	.Execution frequency & timing	RSFFQ		TBD
	.Number of system interfaces serviced by function	RSFIS	Positive Integer	
	.Identifier for each system interface J serviced	RSFII (J)	6 characters	Must match entry in RICID (Group 2)

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	For each function to function interface			
	.Interface identifier	RSMFI	18 characters	Must match system block of group 2
	.Source function identifier	RSMFS	18 characters	Must match entry in RSFID(i)
	.Destination function identifier	RSMFD	18 characters	Must match entry in RSFID(i)
	.Communication Frequency	RSMFC	Positive Integer	Records/second

Table B-2. Candidate Design Parameters

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	DESIGN IDENTIFIERS			
	Design title	AITL	78 characters	For labelling of output reports
	Entry Mode	AIEM	4 character code	
			="MAN"	Manual allocation
			="AUTO"	Automatic allocation
	Design Partition Allocation 1	AIPA	Positive Integer	Used to represent a specific allocation
2	BASIC DESIGN PARAMETERS			
	Number of processors	ANP	I: 1,2,...,MP	
	Number of memories	ANM	I: 1,2,...,MM	
	Number of tasks	ANT	I: 1,2,...,MT	
	Number of blocks	ANB	I: 1,2,...,MB	
	Number of communication links	ANL	I: 1,2,...,ML	
	Number of external I/O devices	ANE	I: 1,2,...,ME	

ISSUE 1 DATE 14-AUG-79 ID DEALS SEC IOPT-9 PAGE 1

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Total number of components = ANP + ANL + ANE + ANM	ANC	I: 1,2,...,MC	
3	CONFIGURATION COMPONENTS			
	For each component c = 1 to ANC (in order by technology category)			
	.Component identifier [ANC]	ACID(c)	6 character mnemonic	ACID(i) .NE. ACID(j) for i.NE.j
	.Technology category [ANC]	ACTC(c)	2 character code	Must match category codes in technology data base IOPT-13 group 2
	.Specific component attributes [ANC,*]	ACCA(c,k)		Dependent upon category in technology data base
	Component connection matrix from i to j [ANC,ANC,2]			

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Primary Link	*ACCM* (i,j,1)	6 characters	Must match candidate design component identifier or left blank
	.Secondary Link	ACCM (i,j,1)	6 characters	Must match candidate design component or left blank
4	DATA BLOCK b's ATTRIBUTES AND ALLOCATION			
	Identifier [ANB]	ABID(b)	6 character mnemonic	*ABID(i)* .NE. ABID(j) for i .NE.j
	Level [ANB]	ABLV(b)	1 character	
			='S'	System interface
			='G'	Global (used by more than 1 task)
			='L'	Local to one task but must be saved
			='T'	Temporary scratch area for a given task

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Discipline [ANB]	ABDC(b)	4 character code	Provides basic I/O requirement for determining suitable memory device allocation
			='FIFO'	queue
			='LIFO'	stack
			='SEQ'	sequential
			='RAN'	random
			='ROR'	ready-only random
			='ROS'	ready-only sequential
			='CBUF'	circular buffer
	Number of memories allocated [ANB]	ABNM(b)	Integer,.GE.1	
	For each allocated memory m			
	.Memory component identifier[AND,ABNM]	ABMI(b,m)	6 characters	Must match memory component identifier ACID of group 2

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Sizing attribute			
	-Block length for memory m	ABSA (b,m,1)	Integer .GE. 1	Dependent on memory
	-Maximum records	ABSA (b,m,2)	Integer .GE. 1	Dependent on memory
	-Average records length	ABSA (b,m,3)	Integer .GE. 1	Dependent on memory
	-Minimum record length	ABSA (b,m,4)	Integer .GE. 1	Dependent on memory
	-Maximum record length	ABSA (b,m,5)	Integer .GE. 1	Dependent on memory
5	TASK t's DEFINITION AND ALLOCATED PROCESSOR(s)			
	Identifier [ANT]	ATID(t)	6 character mnemonic	ATID(i) .NE. ATID(j) for i.NE. j
	Source language		10 character code	Must match entry in the master source language list maintained for current processor technology

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Instruction mix for each instruction type:			
	.Instruction identifier		18 character code	Must match entry in master simulator instruction mix identifiers
	.Execution count			Number of instruction interactions considering looping conditions for average and worst case logic
	-Average		Positive Integer	
	-Worst Case		Positive Integer	
	Number input blocks	ATI(t)	Integer .GE.0	
	Data Retrieval for each task input t			
	.Block level		1 character	See ABLV(b) group 4
	.Block identifier		6 characters	See ABID(b) group 4
	.When		6 character code	

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
			='START'	All records read at first of task before main processing
			='ALONG'	Records processed one at a time
	.Average input		positive integer	Records
	.Minimum input		non negative integer	Records
	.Maximum input		positive integer	Records
	Number output blocks	ATO(t)	Integer .GE. 0	
	Data storage for each task output o			
	.Block level		1 character	See ABLV(b) group 4
	.Block identifier		6 characters	See ABLV(b) group 4
	When		6 character code	

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
			='ALONG'	Records are output via individual processing
			='END'	Records are output just prior to task exit
	.Average output		Positive Integer	Records
	.Minimum output		Non-negative integer	Records
	.Maximum output		positive integer	Records
	Enablement			
	.Type		4 character code	
			='TIME'	Time enabled
			='DATA'	Data enabled
			='SLVD'	Slaved to master task
			='TAD'	Time and/or data enabled

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Frequency 1		real	Iterations per second for time enablement
	.Frequency 2		real	Iterations per second for data enablement
	.Frequency 3		real	Iterations per second for slaved
	Number of allocated processors			
	For each processor p			
	-Component Identifier		6 character	See ACID(c) group 3
	-Task level		Integer .GE. 1	
	-Task slot in level		Integer .GE. 1	
	-Input block memories/devices			To be refined during communcation analysis task
	-Output block memories/devices			

Table B-3. Baseline Load Parameters

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	BASELINE LOAD IDENTIFICATION			
	Identifier	LBID	1Ø characters	
	Title	LBTL	7Ø characters	
	Simulated load start time	LBBEG	Real Ø.LE.LBBEG .LT.LBFIN	Seconds
	Simulated load stop time	LBFIN	Real LBBEG.LT.LBFIN .LE.36ØØ.	Seconds
	Number of master sequences	LBNMS	Positive Integer .LE.75	
	Number of external data arrival methods	LBNDA	Non-negative integer .LE.5Ø	
	Number of data processing cycles	LBNPC	Non-negative integer .LE.5	
	Number of data processing threads	LBNPT	Non-negative integer .LE. 2Ø	

ISSUE 1 DATE 14-AUG-79 ID DEALS SEC IDPT-12 PAGE 1

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
2	STATISTICAL MODEL INITIALIZATION			
	Start time for global statistics	LMGB	Real LBBEG.LE.LMGB	Seconds
	Stop time for global statistics	LMGF	Real LMGB.LT.LMGF .LE.LBFIN	Seconds
	Start time for periodic data collection	LMPB	Real	Seconds
	Delta time between summaries	LMPD	Real	Seconds
	Stop time for periodic data collection	LMPF	Real	Seconds
	Random number generator seed(s)			This will be dependent upon design parameters and selected machine for algorithm implementation

ISSUE 1 DATE 14-AUG-79 ID DEALS SEC IOPT-12 PAGE 2

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
3	MASTER SEQUENCES			
	Sequence start time [LBNMS]	LSST(s)	Real Ø,...,LTFIN	
	Type of sequence [LBNMS]	LSTY(s)	4 Character Code	
			='DATA'	Data arrival sequence
			='CYCL'	Cycle sequence
			='THRD'	Thread sequence
	Sequence Identifier [LBNMS]	LSID(s)	6 Character mnemonic	Must match identifier in list for sequence of type LSTY(s).
4	DATA ARRIVAL d ATTRIBUTES			
	Identifier [LBNDA]	LDAI(d)	6 Characters	LDIS(i) .NE. LDIS (j) for i .NE. j
	Arrival distribution parameters [LBNDA,4]	LDAD(d,i)		

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Distribution type	LDAD(d,1)	character code	
			=E	Equal Distant
			=U	Uniform
			=P	Poisson
			=G	Gaussian
	.Duration interval for arrivals	LDAD(d,2)	Real	Seconds
	.Remaining parameters are dependent on distribution type. All frequencies are in arrivals /second.	LDAD(d,3)	Real	Equal Distant: Frequency
				Uniform:
		LDAD(d,3)	Real	I=3 minimum frequency
		LDAD(d,4)	Real	I=4 maximum frequency

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
				POISSON: I=3 expected frequency I=4 minimum frequency I=5 maximum frequency GAUSSIAN: I=3 expected frequency I=4 standard deviation frequency I=5 limit deviation frequency
	Number of data blocks for which this arrival generation sequence applies [LBNDA]	LDAB(d)	Positive Integer	

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	For each block b:			
	.Identifier	LDABI(d)	1$ character	Must match data block identifier in the candidate design (IOPT-9 group 4)
	.Record type (only if data block is a sort queue)	LDABT(d)	1$ characters	Secondary data block identifier (must match data identifier in the candidate design (IOPT-9 group 4)
	.Data block arrival generation cluster level	LDABL(d)	Positive Integer .LE.LDAB(d)	Used to denote independent versus dependent data generation sequences for given set of blocks. All blocks same level size use one generation sequence for the duration of the sequence.

AD-A101 919 TELEDYNE BROWN ENGINEERING HUNTSVILLE AL SYSTEMS DIV F/G 9/2
ADVANCED MULTIPLE PROCESSOR CONFIGURATION STUDY.(U)
MAY 81 S J CLYMER F33615-79-C-0003
UNCLASSIFIED AFHRL-TR-80-43 NL

2 OF 2
AD A
101919

END
DATE FILMED
8-81
DTIC

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Data block rank r in cluster	LDABR(d)	Positive Integer .LE.LDAB(d)	If the name is the same for all blocks the order of data arrival generated events for the given cluster level is random. If the ranks differ they must be unique and a sequential data arrival generation for each block is to be used in the model in order of rank

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
5	DATA PROCESSING CYCLE c ATTRIBUTES			
	Cycle c's identifier [LBNPC]	LCID(c)	6 characters	LCID(i).NE.LCID(j) for i.NE.j
	Frequency of enablement [LBNPC]	LCFQ(c)	Real	Enablements/second
	Number of independent task to be enabled [LBNPC]	LCNTA(c)	Non negative integer .LE. T	
	List of independent task identifiers task t=1 to LCNIT(c) [LBNPC,LCNIT(c)]	*LCITA(c,t)*	*6 characters*	Each entry in the list must match a task identifier in the candidate design (IOPT-9 group 5)
	Number of independent threads to be enabled [LBNPC]	LCNTH(c)	Non negative integer .LE. LBNPT	
	List of independent thread identifiers for thread k = 1 to LCNTH(c) [LBNPC, LCNTH(c)]	LCITH(c,k)	6 characters	Each entry must match a thread identifier in group 6

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Cycle interrupt/ priority discipline			To be defined based upon communi- cation analysis
6	DATA PROCESSING)THREAD k's ATTRIBUTES FOR k=1 to LBNPT			
	Thread identifier [LBNPT]	LTID(k)	6 characters	LTID(i).NE.LTID(j) for i.NE.j
	Number of tasks in thread [LBNPT]	LTNTA(k)	Positive Integer .GT. 1 .LT. T	
	Task thread order:			
	.Task identifier List of each task t in thread k [LBNPT,LTNTA(k)]	LTTAI(k,t)	6 characters	Each entry must match one of the identifiers in the candidate design task identifier (IOPT-9 group 5) If blank a special sub-thread is as- sumed see para- meter LTTAC(k)

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Level in thread for task enablement (parallel tasks are placed on the same level) [LBNPT, LTNTA(k)]	LTTAL(k,t)	Positive Integer .LE. LTNTA(k)	Levels are enabled in accending order upon completion of all previous level dependencies
	.Completion depency code of task t (this feature permits normal sequencing, splits, temporary splits of threads and/or tasks. However it is important to note that a thread is not permitted to reenter itself either directly or indirectly using this mechanism).	LTTAC(K,t)	Character Code	
			='NODE'	Sequential task node
			='SPLIT'	A new idependent thread is to be enabled
			='END'	No other levels affected
			='TSPLT'	A temporary split is being made to another thread but its completion is necessary prior to given thread level enablement.
	.Completion code attribute a dependent upon completion dependency code	LTTCC (k,t,a)		

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-NODE	LTTCC (k,t,1)	Real	Percent to complete before next level
	-SPLIT and TSPLT	LTTCC (k,t,1)	6 characters	Thread identifier
	-TSPLT and NODE	LTTCC (k,t,2)	Positive Integer	Next level to enable

Table B-4. Technology Data Base Parameters

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	TECHNOLOGY FILE IDENTIFIER	TDBID	20 Characters	
2	MASTER TECHNOLOGY LISTS MASTER CATEGORY LIST			
	Number of current categories	TCNC	Positive Integer=14	
	For each category i=1 to TCNC			
	.Category identifier	TCCI(i)	2 character code	Tentative list includes:
	The first three are the primary areas required for software partioning. The others represent sources or destination for processing I/O and only require as a minimum respective I/O transfer rates, blocking, and protocol interface. Category 15 labeled black box is a catch all category. Other	TCCI(1) TCCI(2) TCCI(3) TCCI(4) TCCI(5) TCCI(6) TCCI(7) TCCI(8) TCCI(9) TCCI(15)	PU CL MM CP CC KB DP MB GE IC	*processor unit *communication line (voice/data) *memory *cockpit instrumentation panels *cockpit controls/switches *keyboard/teletype *display *motion base *"g" equipment *instructor/operator control switches

ISSUE 1 DATE 27-NOV-79 ID DEALS SEC IOPT-5 PAGE 1

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	categories can be added. The particular design evaluation environment must be considered as to what categories and level of data needs to be collected.	TCCI(11)	CM	=TV Camera/Model board
		TCCI(12)	PR	=Printer
		TCCI(13)	CR	=Card reader
		TCCI(14)	BB	=Black box
	MASTER DEVICE LIST FOR CATEGORY i			
	.Number of devices of category i currently in technology data base	TCND(i)	Non negative integer	
	.Device list (for category i) of device identifiers j=1 to TCND(i)	TCDL(i,j)	18 characters	Each entry must be unique within the list for given category i
	MASTER INSTRUCTION LIST			
	.Number of benchmark instructions	TCNI	positive integer	
	.Instruction list for instruction i=1 to TCNI	TCIL(i)	18 character code	TCIL(i) not equal TCIL(j) for i not equal j

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	MASTER BLOCK DISCIP-LINES			
	.Number of disciplines	TCNBD	positive integer=7	
	.List of discipline keys for i=1 to TCNBD in alphanumeric order	TCBDL(i)	4 character codes	
		TCBDL(1)	='CBUF'	circular buffer
		TCBDL(2)	='FIFO'	queue
		TCBDL(3)	='LIFO'	stack
		TCBDL(4)	='RAN'	random I/O
		TCBDL(5)	='ROR'	read only random
		TCBDL(6)	='ROS'	read only sequential
		TCBDL(7)	='SEQ'	sequential I/O
	MASTER BLOCK TYPES			
	.Number of types	TCNBT	positive integer=5	
	.List of block type keys for i=1 to TCNBT in alphanumeric order	TCBTL(i)	1 character code	
		TCBTL(1)	='G'	global
		TCBTL(2)	='I'	instructions
		TCBTL(3)	='L'	local
		TCBTL(4)	='S'	system
		TCBTL(5)	='T'	temporary or scratch

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	MASTER DEVELOPMENT SOURCE LANGUAGES			
	.Number of Languages	TCNL	Positive Integer	
	.List of Language Keys for i=1 to TCNL in alpha-numerical order	TCLL (i)	18 characters	
3	PROCESSOR ATTRIBUTES FOR EACH PROCESSOR P			
	Operating System Features			
	.Multitasking			
	-Levels	TPOSM(p,1)	Integer .GE. 1	Number of distinct task execution levels

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Number of priority service levels	TPOSM(p,2)	Integer .GE. 0 .LE.TPOSM(p,1)	The remaining TPOSM (p,1)-TPOSM(p,2) Levels are assumed to be service in a circular fashions
	.Enablements			
	-Maximum TIme enablement frequency	TPOSM(p,3)	Integer	Enablements/second
	-Resource management per time enablement	TPOSM(p,4)	F10.9 .GE. 0	Seconds accurate to Nano seconds
	-Maximum data enablement frequency	TPOSM(p,5)	Integer	Enablements/second
	-Resource management per data enablement	TPOSM(p,6)	F10.9 .GE. 0	Seconds accurate to Nano seconds
	-Maximum slaved enablement frequency	TPOSM(p,7)	Integer	Enablements/second
	-Resource management per slaved enablement	TPOSM(p,8)	F10.9 .GE. 0	Seconds accurate to Nano seconds

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Resource management overhead per second per task	TPOSM(p,9)	F18.9 .GE. 0	Seconds accurate to Nano seconds
	.For each task level L=1 to TPOSM(p,2)			
	-Maximum number of tasks level L	TPOLT (p,L,1)	Integer .GE. 1	
	-Task service scheme for level L	TPOLS (p,L,2)	Code	
			='P'	Priority
			='C'	Circular
			='F'	First in first out
	.Level resource management	TPOLM (p,L,3)	F18.9 .GE.0	Seconds accurate to Nano seconds
	.List of compatible user memories			
	Simulation instruction set measurements for each benchmark instruction I			
	.Sizing measurements			
	-Number of code memories involved	TPSCM(p,1)		

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	The memory type for each code memory m (the first memory is the user task code--any other memories are predefined for this processor)	TPSMT (p,i,m)	4 character code	Must agree with master memory types defined in Group 4
	-Length of code in memory m	TPSMT (p,i,m,3)	Integer .GE.1	Number of basis units used to desirbe memory m (see Group 4)
	.Timing Measurements for each code memory m and k=1,2			k=1 implies average k=2 implies worst case
	-Number of instruction of scratch data fetch waits	TPTM (p,i,m, 1,k)	Integer .GE. 0	
	-Number of scratch data store waits	TPTM (p,i,m, 2,k)	Integer .GE. 0	
	-Computational total for all memories	TPTT (p,i,k)	Integer .GE. 0	Cycles

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Application development measurements using language L of the master language list			
	-One time development charge	TPDC (p,i,I,L)	Integer	Manhours
	-Change per application instruction of this type	TPDC (p,i,2,L)	Integer	Manhours
4	COMMUNICATION LINE ATTRIBUTES			To be defined in Multiple processor communication analysis task
5	MEMORY ATTRIBUTES FOR MEMORY DEVICE M			
	Type	TMTP(m)	4 characters	
			='ROM'	Read only memory
			='RAMM'	Random access main memory
			='RRAM'	Rotating random access memory
			='SM'	Sequential memory

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
			='WCS'	Writable Control Store
	Number of different addressable units	TMNAU	Positive Integer	
	Size in bits			
	.Min	TMSZ(m,1)	positive integer	bits
	.Max	TMSZ(m,2)	positive integer	bits
	.Increments	TMSZ(m,3)	positive integer	bits
	For each addessable unit u=1 to TMNAU			
	.Level	TMAUP (m,u,1)	4 character code	
			='BIT'	bit addressable
			='6BB'	six bit byte addressable
			='8BB'	eight bit byte addressable
			='WORD'	word addressable
			='HWRD'	half word addressable
			='DBWD'	doubleword addressable

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Bits/unit level	TMAUP (m,u,2)	positive integer	exclusive of parity or error deletion correction bits
	.Read access time	TMAUP (m,u,3)	real	Seconds accurate to nano-seconds
	.Read cycletime per unit	TMAUP (m,u,4)	real	Seconds accurate to nano-seconds
	.Maximum sequential units transferred for single read	TMAUP (m,u,5)	positive integer	same as unit level
	.Write acces time	TMAUP (m,u,6)	real	Seconds accurate to nano-seconds
	.Write cycletime/ unit	TMAUP (m,u,7)	real	Seconds accurate to nano-seconds
	.Max sequential units for single write access	TMAUP (m,u,8)	positive integer	same as unit level
	.Error detection/ correction	TMAUP (m,u,9)	6 character code	
			s'PARITY'	parity bit
			s'SECDED'	single bit error correction double bit error detection

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Number of Suppliers for each supplier	TPMNS(m)	positive integer	
	.Identifier	TPMSP (m,s,1)	18 characters	unique identifier
	.MTBF	(TPMSP (m,s,2)	real	hours-mean time between failures
	.MTTR	TPMSP (m,s,3)	real	hours-mean time to repair
	.MSPM	TPMSP (m,s,4)	real	hours-rescheduled preventative maintenance
	.MTPM	TPMSP (m,s,5)	real	hours-mean time for preventative maintenance
6	COCKPIT INSTRUMENTATION PANEL ATTRIBUTES			
7	COCKPIT CONTROLS/ SWITCHES			
8	KEYBOARD/TELETYPE ATTRIBUTES			
9	DISPLAY ATTRIBUTES			

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
10	MOTION BASE ATTRIBUTES			
11	"G" EQUIPMENT ATTRIBUTES			
12	INSTRUCTOR/OPERATOR CONTROL/SWITCH ATTRIBUTES			
13	TV CAMERA/MODEL BOARD ATTRIBUTES			
14	PRINTER ATTRIBUTES			
15	CARD READER ATTRIBUTES			
16	GENERIC BLACKBOX ATTRIBUTES			

ISSUE 1 DATE 6-AUG-79 ID DEALS SEC IOPT-5 PAGE 12

Table B-5. Evaluation Options

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
1	EVALUATION RUN IDENTIFICATION			
	Evaluation run title	EITLE	characters	28 characters maximum for run identification
	System requirements identifier used to label output and fetch appropriate system requirements file data	EIRI	characters	28 characters maximum. if blank assume value from file otherwise perform equality check
	Baseline time load identifier used to label output and fetch simulation load generation parameters	EIBL	characters	28 characters maximum. if blank assume value from file otherwise perform equality check
	Canadidate design identifier used to label output and fetch appropriate candidate design file data	EICD	characters	28 characters maximum. if blank assume value from file otherwise perform equality check

ISSUE 1 DATE 14-AUG-79 ID DEALS SEC IOPT-14 PAGE 1

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Technology identifier used to label output and fetch appropriate technology data	EITD	characters	28 character maximum. If blank assume value from file otherwise perform equality check
2	EVALUATION OPTIONS			
	Global timing/sizing factors			
	.Cycle iteration factor to apply to all cycle,subcycle, etc. iteration rates	EOCF	F4.2,.GT.0	
	.Processor timing factor	EOPF	F4.2,.GT.0	
	.Memory access time factor	EOMF	F4.2,.GT.0	
	.Data arrival rates factor record length	EODA	F4.2,.GT.0	
	.Data record length sizing factor	EODL	F4.2,.GT.0	
	.Data maximum records factor	EODR	F4.2,.GT.0	
	.Task timing factor	EOTT	F4.2,.GT.0	

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	Selective timing/ sizing			
	.Number of cycles to be modified	EONC	Integer .GE.0	
	.For each cycle c			
	-Identifier [EONC]	EOCI (c)	6 character mneomic	Must match cycle identifier in baseline load IOPT-12 group 5
	-Factor [EONC]	EOCF (c)	F4.2 .GT.0	
	.Number of processor to be modified	EONP	Integer .GE. 0	
	-Identifier [EONP]	EOPI (p)	10 characters	Must match processor device identifier in technology data base IOPT-13 group 2
	-Factor [EONP]	EOPF (p)	F4.2 .GT. 0	

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Number of data block sizing factors	EONDB		
	.For each data block b sizing change			
	-Identifier [EONDB]	EODBI (b)	6 characters	Must match data block identifier in candidate design IOPT-9 group 4
	-Record length factor [EONDB]	EODRL (b)	F4.2, .GT.0	
	-Maximum records factor [EONDB]	EODMR (b)	F4.2, .GT. 0	
	.Number of task timing changes	EONT	Integer .GE. 0	
	.For each task t			
	-Identifier [EONT]	EOTI	6 character mneomic	Must match task identifier in candidate design IOPT-9 group 5
	-Factor [EONT]	EOTF	F4.2, .GT. 0	

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	.Number of memory access times to be modified	EONM	Integer .GE. 0	
	.For each memory m			
	-Identifier [EONM]	EOMI(m)	10 characters	Must match memory device identifier in the technology data base IOPT-13 group 2
	-Read access factor [EONM]	EOMRF(m)	F4.2, .GT. 0	
	-Write access factor [EONM]	EOMWF(m)	F4.2, .GT. 0	
	.Number of data arrival sequences to be modified	EONDA		
	.For each data arrival d			
	-Identifier [EONDA]	EODAI(d)	6 characters	Must match data arrival generation sequence in IOPT-12 group 6 of baseline load
	-Factor [EONDA]	EODAF(d)	F4.2, GT. 0	

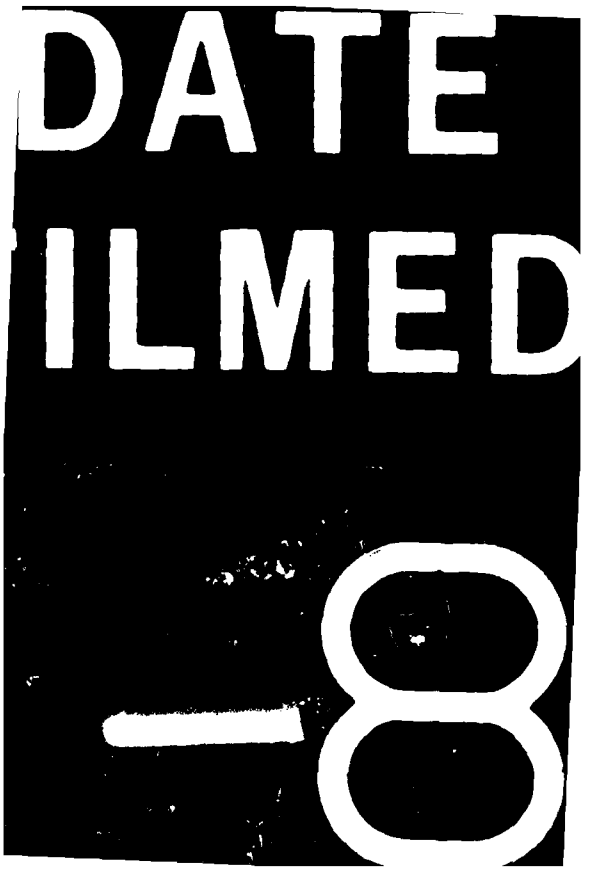
DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
3	REQIORED MEASURES			
	Global Component Utilization Spare			
	.Processor Factor	ERGPF	F4.3, .GE. 0 .LT. 1	
	.Memory Factor	ERGMF	F4.3, .GE. 0 .LT. 1	
	.Communication link factor	ERGLF	F4.3, .GE. 0 .LT. 1	
	.Cycle Factor-portion of length between enablements	ERGCF	F4.3, .GE. 0 .LT. 1	
	Selective component utilization			These override global factors
	.Number of selected components	ERNSC	Integer .GE. 0 .LE.C	
	.For each component c			
	-Technology category (ERNSC)	ERSCC(c)	2 characters	

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
			='PU'	Processor Unit
			='MM'	Memory
			='CL'	Communication Link
	-Component identifier	ERSCI(c)	6 characters	Must match component identifier for given category in candidate design IOPT-9 group 3
	-Component space factor	ERSCF(c)	F4.3, .GE. 0 .LT. 1	
	Thread end-to-end timing collection			
	.Number of threads for which data is to be collected	ERNK	Integer .GE. 0 .LE. K	
	.For each thread k			
	-Identifier [ERNK]	ERKI(k)	6 character mnemonic	Must match thread identifier of baseline load IOPT-12 group 6

DEALS

GRP	PARAMETER NAME	MNEMONIC	VALUES	UNITS/VALUE MEANING
	-Maximum time limit factor [ERNK]	ERKT(k)	F4.3, .GT. 0	
	Task end-to-end timing collection			
	.Number of tasks for which data is to be collected	ERNT	Integer .GE. 0 .LE. T	
	.For each task t			
	-Identifier [ERNT]	ERTI(t)	6 character mnemonic	Must match task identifier in candidate design IOPT-9 group 5
	-Maximum time limit factor [ERNT]	ERTT(t)	F4.3, .GT. 0	


DATE
ILMED
8