NASA Technical Memorandum 83605

Operating System for a Real-Time Multiprocessor Propulsion System Simulator

Gary L. Cole
Lewis Research Center
Cleveland, Ohio

Prepared for the
Summer Computer Simulation Conference
sponsored by the Society for Computer Simulation
Boston, Massachusetts, July 23–25, 1984

NASA

OPERATING SYSTEM FOR A REAL-TIME MULTIPROCESSOR PROPULSION SYSTEM SIMULATOR

Gary L. Cole

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

The NASA Lewis Research Center is developing and evaluating experimental hardware and software systems to help meet future needs for real-time, high-fidelity simulations of air-breathing propulsion systems. Specifically, the Real-Time Multiprocessor Simulator project focuses on the use of multiple microprocessors to achieve the required computing speed and accuracy at relatively low cost. Operating systems for such hardware configurations are generally not available. This paper describes the Real-Time Multiprocessor Operating System (RTMPOS) that has been developed at NASA-Lewis. The RTMPOS provides the user with a versatile, interactive means for loading, running, debugging and obtaining results from a multiprocessor-based simulator. A front-end processor (FEP) serves as the simulator controller and interface between the user and the simulator. These functions are facilitated by the RTMPOS which resides on the FEP.

The RTMPOS acts in conjunction with the FEP's manufacturer-supplied disk operating system that provides typical utilities like an assembler, linkage editor, text editor, file handling services, etc. Once a simulation is formulated, the RTMPOS provides for engineering-level, run-time operations such as loading, modifying and specifying computation flow of programs, simulator mode control, data handling and run-time monitoring. Run-time monitoring is a powerful feature of RTMPOS that allows the user to record all actions taken during a simulation session and to receive advisories from the simulator via the FEP. The RTMPOS is programmed mainly in PASCAL along with some assembly language routines. The RTMPOS software is easily modified to be applicable to hardware from different manufacturers.

INTRODUCTION

Simulations of complex dynamic systems require a versatile interface between the user and the simulation computer. In order to maximize the usefulness of the simulation as an engineering tool, the interface should provide a user-friendly means of programming, interacting with and obtaining results from the simulation. In the past, different types of computers have satisfied some but not all of these requirements. For example, the use of analog computing equipment provides immediate results and an extremely versatile user/simulator interactive capability. But programming and changing of the program can be difficult. On the other hand, digital computers are more easily programmed but their operating systems are generally not designed to provide the interactive capabilities needed for many simulation tasks.

Simulations are becoming more sophisticated in terms of the complexity of the systems being modeled and in the level of model details being sought. Therefore, the computer hardware and associated operating systems that are needed to support these simulation efforts, in real time, are also becoming more complex. An example is the Real-Time Multiprocessor Simulator (RTMPS) being studied at the NASA Lewis Research Center. The objective of the RTMPS project is to develop and evaluate experimental hardware and software systems for real-time interactive simulation of air-breathing propulsion systems. The RTMPS project is focusing on the use of multiple microprocessors to achieve the required computing speed and accuracy at low cost relative to hybrid and main frame digital computers.

University grant and contract efforts related to the RTMPS project are described in Refs. 1 to 4. These include investigations of computer architecture and related hardware requirements (1), approaches to partitioning simulation models for solution on multiple microprocessors (2,3) and requirements for high-level programming languages for multiple processor systems (4).

The NASA-Lewis in-house efforts, to date, are documented in this paper and in Refs. 5 and 6. The general multiprocessor simulator concept is described in Ref. 5. The experimental hardware configuration now in use at Lewis is described in this paper and in Ref. 6 Reference 6 also describes a high-level Real-Time Multiprocessor Programming Language (RTMPL) developed at NASA-Lewis.

Although the RTMPS hardware is complex, it is anticipated that the desired user/simulator interface capabilities can be achieved through proper design of the operating system. The purpose of this paper is to give an overview of the design and capabilities of the Real-Time Multiprocessor Operating System (RTMPOS). The RTMPOS is designed to allow highly-interactive, engineering-level programming and operation of multiprocessor systems. It is designed to be expandable and to minimize dependence on specific manufacturer hardware. In the case of the Lewis experimental RTMPS, the RTMPOS provides special system tasks that operate in conjunction with a standard, manufacturer-supplied disk operating system (DOS). This paper describes the multiprocessor architecture that is supported by the RTMPOS, the general programming and operational functions that are provided, and the RTMPOS structure that supports those functions.

GENERAL SIMULATOR CONFIGURATION

The Real-Time Multiprocessor Programming Language (RTMPL, ref. 6) and the Real-Time Multiprocessor Operating System described in this paper are designed to support the general simulator configuration shown in Fig. 1. The primary elements in this configuration are the multiple simulation channels (1 to n), the Front-End Processor (FEP), and the Real-Time Interface. The Real-Time Interface provides for communications between the simulator and external devices (e.g. strip-chart recorders and controls).

The FEP serves as the simulator controller and the interface between the user and the simulator. These functions are facilitated by the RTMPOS which resides on the FEP. Data are transferred between the FEP and the simulation channels via the Interactive Information Bus. The RTMPOS provides for simulator run-time operations such as program loading and modification, simulator mode control, and data handling. The FEP also services the simulator peripherals (terminals, disk drives, printers, etc.). A manufacturer-supplied DOS provides typical utilities like an assembler, linkage editor, text editor, file handling services, etc. The resident DOS has a multi-tasking capability (i.e. permits many tasks to run concurrently in a time-slice mode). This capability is essential to the RTMPOS concept since the RTMPOS includes several tasks that permit the simulator to send advisories to the user via the FEP while the simulation is being run.

Each simulation channel (fig. 1) consists of two processors--a computation processor (COMP) and a preprocessor (PREP). Each COMP executes its assigned portion of the simulation (an RTMPL program) and serves to interface its channel to the FEP. The PREP's also execute RTMPL programs and provide for distribution of information from the COMP's to the other channels via the Real-Time Information Bus. Communication between the COMP and PREP is accomplished through a shared memory. Each simulator processor contains specially-designed firmware (software burned into EPROMs) that facilitates communication between the FEP and the COMP's and between the COMP's and The PREP's. The firmware performs such functions as initialization and checkout of the processor memory, data transfer, setting the execution mode of the processors, sending interrupts out to the FEP and, in some cases, timer control.

One of the simulation channels (1 to n) is different from the others. It serves as a real-time extension of the FEP. Its COMP is available to perform any real-time analysis required by the user to support the simulation. For example, it may be used to gather and process data from the other channels which are then uploaded to the FEP by means of the RTMPOS. The real-time PREP is used to distribute data on the Real-Time Information Bus and serves as controller for that bus. It is also responsible for all timing and control of simulator operations.

The communication paths that are available in the general simulator configuration provide a high degree of programming flexibility. As a result, the RTMPS implementation may be used to emulate a variety of multiprocessor systems. This flexibility also means that the RTMPOS design can be viewed as generic to a variety of multiprocessor systems that are subsets of the general configuration shown in Fig. 1.

Figure 2 is a photograph of the current NASA-Lewis RTMPS experimental hardware, the FEP, and the peripheral equipment. The NASA-Lewis RTMPS uses a Motorola EXORmacs* Development System as the FEP and the resident DOS is Motorola's VERSAdos (7)*. The FEP and simulator processors are based on the Motorola MC68000 microprocessor with an 8 MHz clock.

FORMULATING A SIMULATION

An RTMPS simulation is formulated using the RTMPL utility (6). The RTMPL allows the user to program the various elements of the general simulator configuration in a high-level, engineering-oriented language. As shown in Fig. 3, the output of the RTMPL is assembly language source program files (one for each processor that is being used in the simulator) and a set of simulation data base files that relate the simulation implementation to the RTMPL source programs.

Each assembly source program includes items such as executives, tasks, variables, constants, and argument groups. Executives and tasks define the executable part of the simulation. An executive is like a main program that directs the computational flow of that processor's simulation program. Tasks are used like subroutines to partition a program to improve its readability and versatility. An executive may perform tasks itself. Each program must have at least one executive but tasks are not required. Constants and variables are elements in the simulation equations that are to be executed in the program. Variables are typically time-dependent variables such as would appear in a differential equation. Because of the parallel nature of the solution, some variables must be transferred to/from other processors. An argument group is a set of constants and/or variables that are grouped under a single name for ease of reference. Argument groups provide for large-volume data transfers between the FEP and the simulator. One application would be to periodically pass an argument group to a sampling routine to obtain simulation results.

The program load modules that are loaded into the simulator processors are generated by passing the source programs through the FEP-resident assembler and linkage editor.

The data base files that are generated by the RTMPL are files of records that are read by the RTMPOS. These files contain all information about the simulation that is necessary for the RTMPOS to support interactive execution of the simulation. There are files of information relative to each processor program as well as files of globally-defined items that can be referenced by all simulation programs. The kind of information typically included in the data base is illustrated by the constant record shown in Fig. 4. Constants are stored in linked lists (hence the pointer "next") of records that include the constant's name (ID), location in processor memory (LOC), and its DTP--data type (scaled fraction or integer) and precision (number of words in memory). This information allows the user to reference a constant (or other item) by name and allows the RTMPOS to reference it in memory.

*EXORmacs and VERSAdos are trademarks of Motorola Inc.

All RTMPL data base and program source/object-code/load-module files are saved on disk files that can be manipulated by the RTMPOS. Once the simulation has been formulated, the RTMPOS provides the following general programming and operational functions·

- Program Control
- Data Base Management
- Simulator Control
- Run-Time Monitoring
- Simulation Results Management
- Miscellaneous

DESCRIPTION OF RTMPOS FUNCTIONS

Data Base Management

The first step in a simulation session is to load the data base from the RTMPL-generated disk files into the FEP memory. The RTMPOS then uses the data base to allow the user to interactively modify and execute the simulation. The data base management functions provide for

- Loading
- Editing
- Saving
- Listing

of the data base. Editing refers to making changes to the simulation at run time such as displaying/-changing values of constants and initial condition (IC) values of variables. Advisory messages may be changed and additions, deletions or changes to items contained in argument groups may be made. The data base may be edited before or after the simulation program load modules have been loaded into the simulator. In either case the simulation programs are automatically updated by the RTMPOS to be consistent with the changes in the data base. The original data base and load modules on the disk files remain unmodified. An edited data base may be saved by the user either by overwriting the original disk files or by creating a new set.

The RTMPL utility generates a listing of the entire data base. Once the data base is loaded into the FEP, the user has the option of listing all or selected portions of the original or edited data base.

Program Control

At run time the RTMPOS program control functions are used for loading the program load modules from the disk files into the desired simulator channels, activating the desired executive in each program, and enabling or disabling tasks in the executive. The use of executives and tasks in structuring a simulation program can provide the user with a great deal of flexibility at run time. The status of program executives and tasks can be reviewed and changed by the user at any time.

Simulator Control

The user controls the simulator in much the same manner as an analog computer. Three modes of execution are available

- STOP
- RUN
- HOLD.

In the STOP mode the simulator is halted and simulation execution is suspended. Changes to the data base are allowed only in STOP. While in STOP, the user can specify which simulation variables are to have their IC or hold values set. IC and hold values are specified in the data base and they can be changed by using the data-base editing commands.

In the RUN mode all simulation programs execute repeated update cycles. The simulator stays in RUN until the user executes the STOP command or the simulation itself issues a halt advisory. The halt advisory may be a user generated message such as "SIMULATION HALT! MAX TEMPERATURE EXCEEDED" or a system advisory such as "SIMULATION HALT! DIVIDE BY ZERO IN CHANNEL 2 PREP". All advisories are displayed on a user-defined message device (e.g., a printer).

In the HOLD mode, user-selected simulation variables in each program are held at the specified values. After selecting the variables, the user specifies the number of update cycles to be executed. The simulation executes the update cycles and then returns to the STOP mode and issues a "HOLD MODE COMPLETE" advisory to the user.

Simulator control also includes initialization of simulator hardware and user-specification of timer intervals that control the execution of the simulation (e.g., update time).

Run-Time Monitoring

This function allows a user to receive advisories via interrupts from the simulator and to record all commands that are executed and the resulting actions that are taken during a simulation session. The advisories can be user or operating-system messages or they can be requests to read data from the simulator. The messages are displayed on the message device. The read advisories are built into the simulation by the user. They are transparent to the user at run time and are automatically serviced by special RTMPOS tasks.

A powerful feature of run-time monitoring is the self-documenting session history--a disk text file that saves all user commands and resulting prompts and messages from the RTMPOS. Any advisory messages that occur are automatically entered into the session history, as are user-generated comments. The session history may be listed and edited using the manufacturer-supplied utilities that are resident on the FEP. Furthermore, a session history may be executed by the RTMPOS. That is, commands can be input to the RTMPOS from a session history file rather than being entered manually from the keyboard. This allows the user to quickly bring a simulation to a previously obtained condition by executing the session history from that session. The user may also create a file with the text editor that can be used to execute the routine parts of a simulation session (e.g., loading a data base and the simulator load modules). After the execution of the session history is completed control is returned to the user at the keyboard.

Results Management

Simulation results are obtained by means of the read advisory that was mentioned in the previous section. The user includes requests for sampling of argument groups in the RTMPL source program. When the simulation is running and a read advisory is encountered, an interrupt is sent out to an appropriate RTMPOS task. Each channel has its own task. The RTMPOS task transfers the argument group data from the simulator to an auxiliary memory block in the FEP. Whenever the simulator enters the STOP mode, the RTMPOS converts the data to the proper units using data base information, enters the data with identification into a user-defined text file, and clears the auxiliary memory. The data are coordinated with the session history file by means of a run number and an RTMPOS comment in the session history that indicates the number of data records that were saved. The data may be listed using the FEP-resident list utility.

Future enhancements in the results management area are expected to include graphical display of data and user-defined means for data analysis.

Miscellaneous

Once the RTMPOS has been invoked by the user there are two ways to exit from it. One command results in all RTMPOS tasks being permanently terminated. A different command allows the user to temporarily suspend the RTMPOS command task, perform other tasks such as editing and listing a text file and then resume the RTMPOS at the point that the user left off. In the latter case, the simulator can continue to run with any advisory messages output to the message device to keep the user informed of the simulation's progress. A "help" command is also available that permits the user to view the RTMPOS commands and a brief description of their functions.

RTMPOS STRUCTURE

A rather complex software structure is required to provide all of the RTMPOS functions that have been described. The overall RTMPOS structure and its relationship to the FEP/simulator is shown in Fig. 5. As explained earlier, the RTMPOS resides on the FEP (EXORmacs) and operates in conjunction with VERSAdos. The RTMPOS interacts with the EXORmacs utility software and with the peripheral equipment by means of the file-handling (FHS) and the input/-output (IOS) services. The RTMPOS block of Fig. 5 is shown in more detail in Fig. 6 to illustrate the multi-task design of the RTMPOS. The major RTMPOS tasks are the Command task and the tasks associated with servicing the advisory interrupts that come from the simulator. An auxiliary memory (131k bytes) is used to provide a communication link between the tasks as well as storage for advisory messages and simulation-data results.

The RTMPOS command task processes all of the user-entered commands. The user enters a command for the desired function and is then prompted by the RTMPOS for subsequent inputs. This task consists of mostly PASCAL procedures with some assembly language routines. The assembly language routines are used mainly for moving information between the command-task memory and other (auxiliary and COMP) memories. The assembly routines are specific to the MC68000 microprocessor but should be easy to adapt to other manufacturers' hardware. The PASCAL procedures are generic except for the naming of files (VERSAdos-specific). These procedures should also be easy to adapt to other manufacturers' file-handling systems. The PASCAL structured approach to programming facilitated the design of the RTMPOS and should also facilitate any additions and/or modifications to the RTMPOS. The present version of the RTMPOS command task requires approximately 210k bytes of memory. This includes the memory required to support the data base for a relatively small 3-channel simulation.

As shown in Fig. 6, each channel is assigned an interrupt service task (1.5k byte), a message advisory task (5k byte) and a read advisory task (1k byte). After receiving an interrupt, the interrupt task determines the advisory type and then initiates the appropriate advisory task. Each of these tasks is written in assembly language and is specific to the MC68000 microprocessor.

Enough auxiliary memory is reserved for 95 messages. If the allotted memory is exceeded the user is notified by a system message and the oldest messages are overwritten. One hundred twenty-nine kilobytes of auxiliary memory are reserved for data. This is sufficient to save approximately 2500 argument group records, each containing 20 single-precision items. If the allotted memory is exceeded the user is notified by a system message. The simulator continues to run but no additional data records are saved.

One additional task that is not shown in Fig. 6 is a small (500 byte) assembly routine that is used to temporarily suspend operation of the command task. The user can then use the FEP for other purposes (with or without the simulator running). The user can resume operation of the RTMPOS command task and interactive operation of the simulator at any time.

At the present time the RTMPOS is designed to accommodate a maximum of 10 simulator channels. The NASA-Lewis experimental RTMPS hardware contains only 3. For a 10-channel system the RTMPOS would consist of 32 tasks requiring a total memory capacity of approximately 0.5 megabytes, including the memory required for a large data base and the auxiliary memory.

CONCLUDING REMARKS

The RTMPOS described in this paper meets the goals of providing a user-oriented, interactive programming and operations environment for a real-time multiprocessor simulator. The RTMPOS acts in conjunction with a manufacturer-supplied disk operating system to provide the desired capabilities. A simulation is first formulated using a high-level programming language. The RTMPOS then provides for run-time operations such as loading, modifying and specifying computational flow of programs, simulator mode control and run-time monitoring. Run-time monitoring includes communication of simulator advisories to the user and recording of a session history file. The session history file is a powerful feature that records all user commands, pertinent RTMPOS prompts and messages and advisory messages from the simulator. The user can use the session history file to review a session's progress without having to take notes and to input commands to the RTMPOS rather than entering them manually through the keyboard.

The RTMPOS functions that are described in this paper represent an initial design that has been programmed and debugged. Testing of this design with the experimental RTMPS hardware was about to begin at the writing of this paper. It is expected that enhancements and additions will be made in the future to allow user-defined data analysis routines and graphical display of simulation results. Since the RTMPOS is part of a research project, the required improvements and their implementation will depend upon the experience gained during the testing phase. The PASCAL structured approach to programming of the RTMPOS should allow modifications and additions to be made with relative ease.

REFERENCES

1. O'Grady, E. Pearse and Wang, Chung-Hsien, "Multibus-Based Parallel Processor for Simulation," Proceedings of the 1983 Summer Computer Simulation Conference, Society for Computer Simulation, La Jolla, CA, 1983, pp. 371-375.

2. Daniele, Carl J. and McLaughlin, Peter W., "The Real-Time Performance of a Parallel, Nonlinear Simulation Technique Applied to a Turbofan Engine," Modeling and Simulation on Microcomputers· 1984, edited by Ray Swartz, Society for Computer Simulation, La Jolla, CA, 1984, pp. 167-171.

3. Makoui, Ali and Karplus, Walter J., "Data Flow Methods for Dynamic System Simulation A CSSL-IV Microcomputer Network Interface," Proceedings of the 1983 Summer Computer Simulation Conference, Society for Computer Simulation, La Jolla, CA 1983, pp. 376-382.

4. Collins, W. Robert and Feyock, Stefan, "The Use of ADA in Distributed Simulations," Proceedings of the 1983 Summer Computer Simulation Conference, Society for Computer Simulation, La Jolla, CA, 1983, pp. 364-370.

5. Blech, Richard A. and Arpasi, Dale J., "An Approach to Real-Time Simulation Using Parallel Processing," NASA TM-81731, 1981.

6. Arpasi, Dale J., "RTMPL A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulation," NASA TM-   , 1984.

7. Glaser, Jay G., "The VERSAdos Operating System," Microprocessor Operating Systems, edited by John Zarrella, Microcomputer Applications, Suisun City, CA, 1981, pp. 9-1 to 9-20.

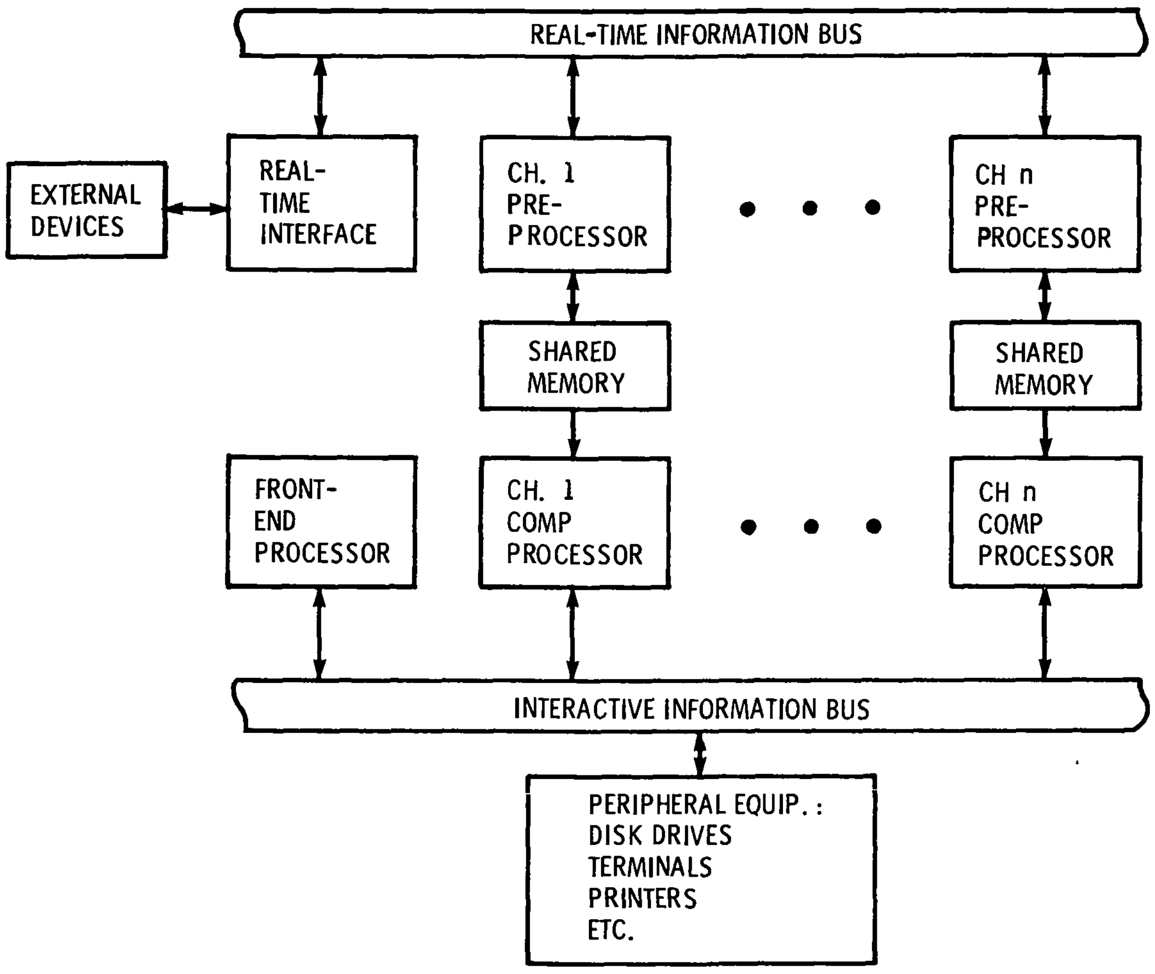

Figure 1. - General simulator configuration.

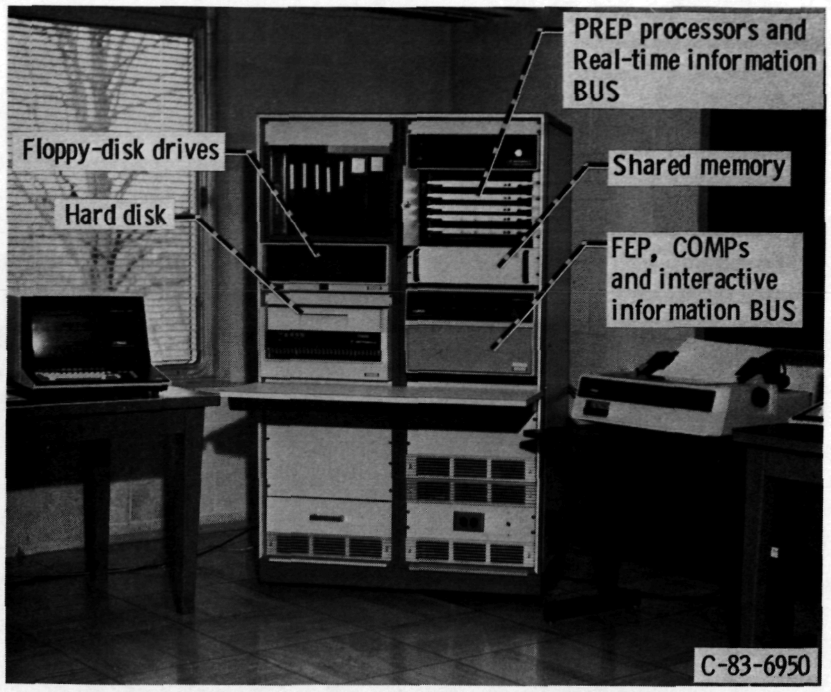
PREP processors and Real-time information BUS
Floppy-disk drives
Shared memory
Hard disk
FEP, COMPs and interactive information BUS
C-83-6950

Figure 2. - NASA-Lewis RTMPS experimental hardware.

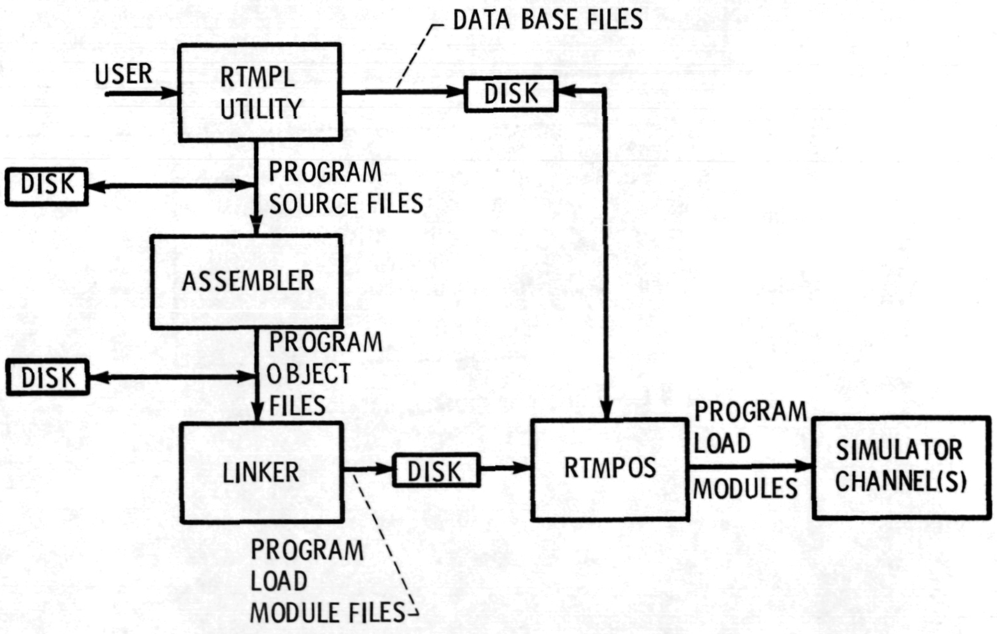

Figure 3. - RTMPL - generated simulation files.

NEXT:	POINTER;
ID:	NAME;
LOC:	STARTING MEMORY LOCATION;
MEM:	NUMBER OF MEMORY LOCATIONS;
DTP:	DATA TYPE AND PRECISION;
SF:	SCALE FACTOR (IF SCALED FRACTION);
PRM:	PARAMETER (ADJ. OR FIXED) T OR F;
SIZE:	ARRAY DIMENSION;
VAL:	POINTS TO VALUE RECORD

Figure 4. - Details of data-base constant record.

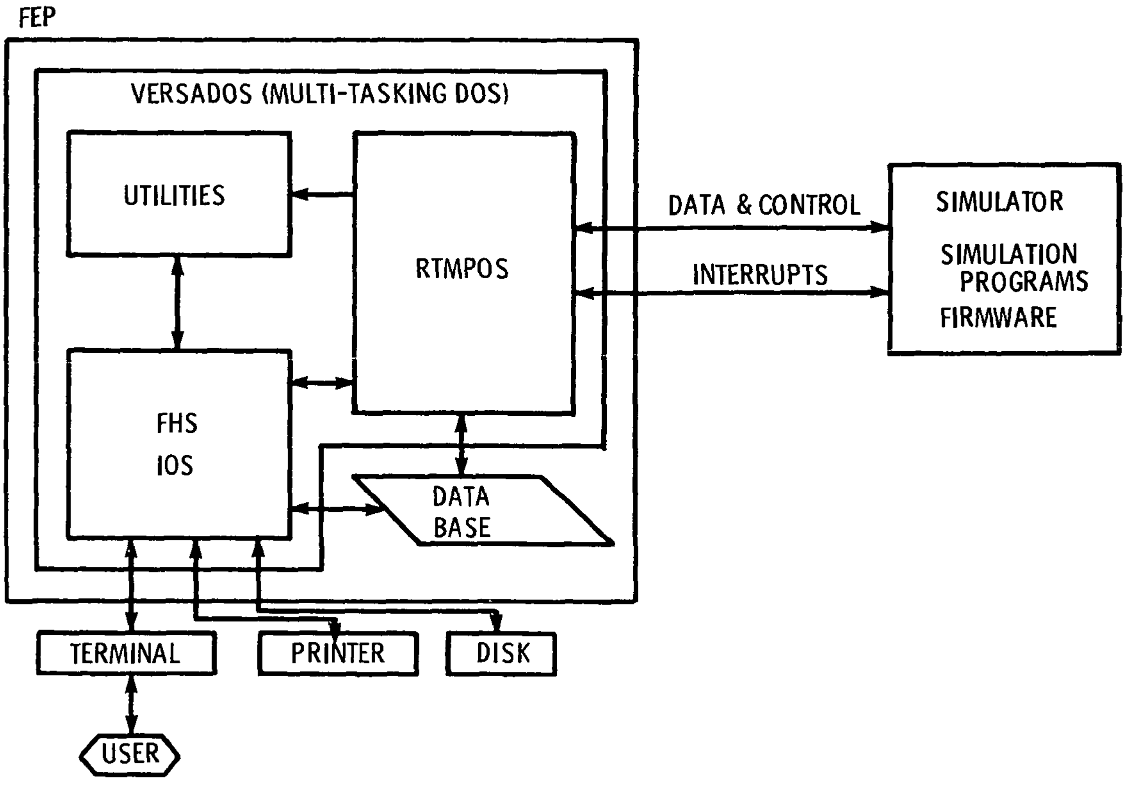

Figure 5. - Overall FEP /simulator software structure.

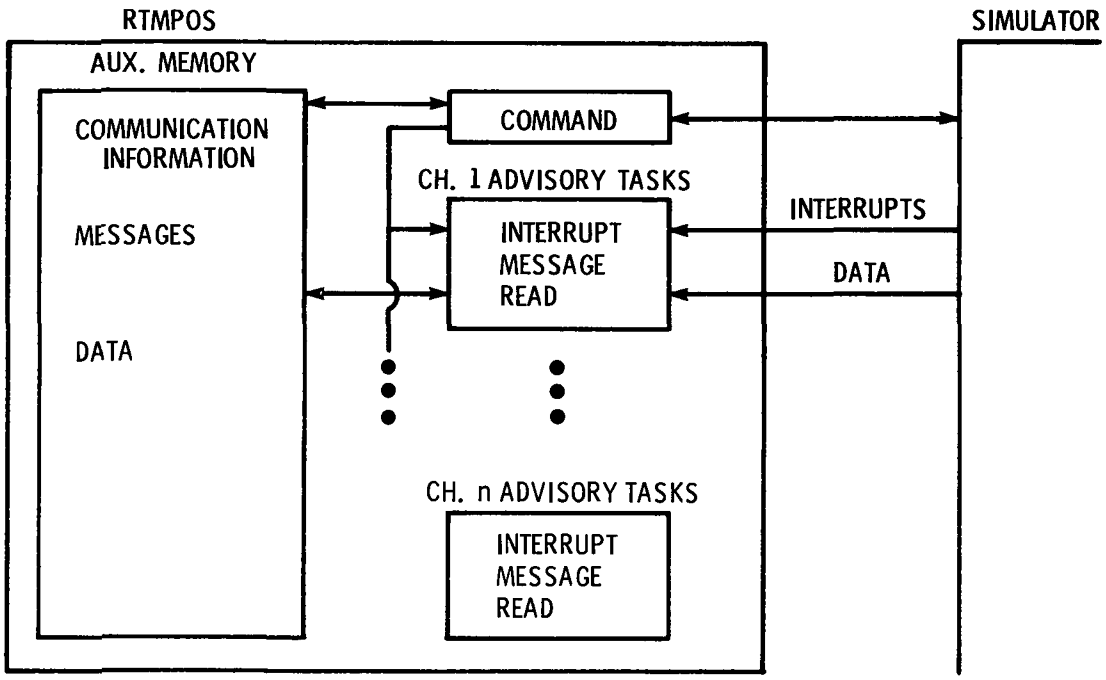

Figure 6. - RTMPOS task structure.

1. Report No NASA TM-83605	2. Government Accession No	3 Recipient's Catalog No
4 Title and Subtitle Operating System for a Real-Time Multiprocessor Propulsion System Simulator		5 Report Date
		6 Performing Organization Code 505-40-5B
7 Author(s) Gary L. Cole		8 Performing Organization Report No E-2023
		10 Work Unit No
9 Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135		11 Contract or Grant No
		13 Type of Report and Period Covered Technical Memorandum
12 Sponsoring Agency Name and Address National Aeronautics and Space Adminstration Washington, D.C. 20546		14 Sponsoring Agency Code

15 Supplementary Notes

Prepared for the Summer Computer Simulation Conference sponsored by the Society for Computer Simulation, Boston, Massachusetts, July 23-25, 1984.

16 Abstract

The NASA Lewis Research Center is developing and evaluating experimental hardware and software systems to help meet future needs for real-time, high-fidelity simulations of air-breathing propulsion systems. Specifically, the Real-Time Multiprocessor Simulator project focuses on the use of multiple microprocessors to achieve the required computing speed and accuracy at relatively low cost. Operating systems for such hardware configurations are generally not available. This paper describes the Real-Time Multiprocessor Operating System (RTMPOS) that has been developed at NASA-Lewis. The RTMPOS provides the user with a versatile, interactive means for loading,running, debugging and obtaining results from a multiprocessor-based simulator. A front-end processor (FEP) serves as the simulator controller and interface between the user and the simulator. These functions are facilitated by the RTMPOS which resides on the FEP. The RTMPOS acts in conjunction with the FEP's manufacturer-supplied disk operating system that provides typical utilities like an assembler, linkage editor, text editor, file handling services, etc. Once a simulation is formulated, the RTMPOS provides for engineering-level, run-time operations such as loading, modifying and specifying computation flow of programs, simulator mode control, data handling and run-time monitoring. Run-time monitoring is a powerful feature of RTMPOS that allows the user to record all actions taken during a simulation session and to receive advisories from the simulator via the FEP. The RTMPOS is programed mainly in PASCAL along with some assembly language routines. The RTMPOS software is easily modified to be applicable to hardware from different manufacturers.

17. Key Words (Suggested by Author(s)) Operating systems; Multiprocessors; Real-Time simulation; Interactive simulation; Simulators	18. Distribution Statement Unclassified - unlimited STAR Category 62

19. Security Classif (of this report) Unclassified	20 Security Classif. (of this page) Unclassified	21 No of pages	22 Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22161