NASA Technical Memorandum 83606

# RTMPL—A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulations

Dale J. Arpasi
*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

# RTMPL - A STRUCTURED PROGRAMMING AND DOCUMENTATION UTILITY FOR REAL-TIME MULTIPROCESSOR SIMULATIONS

Dale J. Arpasi
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 simulations of air-breathing propulsion systems. The Real-Time Multiprocessor Simulator (RTMPS) project is aimed at developing a prototype simulator system that uses multiple microprocessors to achieve the desired computing speed and accuracy at relatively low cost. Software utilities are being developed to provide engineering-level programming and interactive operation of the simulator.

Two major software development efforts were undertaken in the RTMPS project. A real-time multiprocessor operating system (subject of a companion paper) was developed to provide for interactive operation of the simulator. The second effort (the subject of this paper) was aimed at developing a structured, high-level, engineering-oriented programming language and translator that would facilitate the programming of the simulator.

The Real-Time Multiprocessor Programming Language (RTMPL) allows the user to describe simulation tasks for each processor in a straightforward, structured manner. The RTMPL utility acts as an assembly language programmer, translating the high-level simulation description into time-efficient assembly language code for the processors. The utility sets up all of the interfaces between the simulator hardware, firmware, and operating system. All required interprocessor communications (i.e., data transfers) are automatically established by the utility. RTMPL simulations are self-documenting since the utility produces listings, error messages, warnings, and database files that can aid in the debugging and running of a simulation.

The RTMPL utility is written in Pascal. The current implementation runs on a Motorola EXORmacs Development System (host for experimental simulator) and produces MC68000 assembly language code. The utility can easily be retargeted for different processors by changing the contents of a target definition file. The RTMPL language (i.e., command and operation set) is macro-based and can be modified/expanded to meet particular simulation needs.

## INTRODUCTION

A Real-Time Multiprocessor Programming Language (RTMPL) has been designed to aid in developing multiprocessor simulation programs. The language and its utility (translator) are targetable to specific simulator architectures, processor-types, and simulation requirements. To aid in portability of RTMPL, the utility is written, for the most part, in Pascal. Two simple MC68000 assembly language routines are used for special system interfacing. The RTMPL utility functions as an assembly language programmer, accepting an engineering-level description of the simulation and translating it into time-efficient assembler source code. The utility provides program optimization and diagnostic information to the user via output listings. It also provides operational information to the simulator operating system in the form of database files to facilitate real-time, interactive, running of the simulation.

The RTMPL was developed initially to support the development of a Real-Time Multiprocessor Simulator (RTMPS) at LeRC. It was felt that rapid turnaround of code generation would be required in the process of testing and evaluating the simulator and associated simulation techniques. In the long term it is hoped that the demonstrated value of RTMPL will help to promote the acceptance and use of real-time multiprocessor simulation by the general engineering community. This paper presents an overview of the RTMPL language and utility. The general simulator configuration to which RTMPL is applicable is described with emphasis on the simulator's information-transfer capabilities. RTMPL is described in the context of the overall simulation development effort. The targeting of code and the major utility functions are discussed. A brief discussion of the language is included to show how it reflects the general RTMPS simulation philosophy. Finally, a description of the utility's output listing is presented and its usefulness for simulation optimization and execution is discussed. A more detailed description of RTMPL will be provided in a detailed user's manual (to be published). It includes a full definition of the language constructs and presents an example simulation with a complete set of source and output listings.

### General Simulator Configuration

The RTMPL language and utility were developed to provide a means for programming a multiprocessor simulator having the general configuration shown in Fig. 1. Subsets of this configuration are also supported. The general simulator consists of a number of simulation channels. Each channel would be directed to solving a portion of the simulation. There are two special-purpose devices: (1) a

user-interface or front-end processor (FEP) which provides for user interaction with the simulator, and (2) a real-time interface which provides for communications between the simulator and external devices (e.g., controllers, actuators). Internal simulator communications are accomplished via two data paths. The Interactive Information Bus links the FEP and simulation channels. The Real-Time Information Bus links the real-time interface and simulation channels. For noninteractive simulations, both buses can be used for real-time interchannel data communications, thus providing the potential for significant speed improvements relative to a single-bus configuration. However, for interactive simulation, the Interactive Information Bus could be tied up servicing user requests and would not be available for real-time data transfers.

Each simulation channel in the general configuration consists of two processors· a preprocessor tied directly to the real-time bus and a computational processor tied directly to the interactive bus. These processors communicate through shared memory.

The general configuration allows a simulation program to be segmented into 1 to N parts where N is the number of channels. The computations for each channel may be further segmented between the computational and preprocessors. In this way, each channel's computations can be speeded up, relative to a single processor.

The RTMPL utility automatically sets up the appropriate data transfers between and within the channels. It does this using information contained in operands that appear in the RTMPL source programs (one for each processor). Transfer paths can be specified in the source programs allowing any processor to communicate with any other processor via any information path. These features of RTMPL allow the user to generate simulations that will run on any subset of the general simulator. Subsets of the general configuration are the results of eliminating processors and/or data paths from the general configuration. RTMPL programs would be targeted to a particular configuration based on hardware availability or simulation requirements.

## RTMPL Utility

The RTMPL utility functions under a disk operating system. It is presently running under Motorola's VERSAdos on an EXORmacs development system. The utility is specific to a particular system only in the file identification format. The utility is one of a number of utilities (either developed at LeRC or furnished with VERSAdos) which are currently being used for RTMPS simulation development and execution. Figure 2 illustrates the RTMPS utilities and how they are related. Simulation development begins with the system-supplied editor which is used to develop RTMPL source files. These files define the simulation problem and contain programs for each simulator processor. The RTMPL utility translates the RTMPL source into assembler-source files (according to information contained in user-specified target definition files that are described later). The system supplied assembler and linker utilities then produce load modules which are loaded into the simulation processors (MC68000's in the RTMPS) by the Real-Time Multiprocessor Operating System (RTMPOS) at run-time. RTMPOS is a LeRC-developed utility and is the subject of a companion paper (1). The RTMPL utility also produces simulation-descriptive data-base files for use by RTMPOS in providing run-time initialization and interactive operational capabilities. Finally, an RTMPL listing file is provided which contains messages and source interpretations to aid the user in developing error-free, time-optimum simulations.

As mentioned previously, the RTMPL utility makes use of target definition files to perform its translation function. These files provide descriptions of the simulator configuration, target processors, the target assembler, and simulator logical states. Examples of processor information contained in the target definition files are register specifications (number, length, and names), available memory (starting locations and amount), and value representations (single, double, or triple precisions, and values used to represent the logical constants TRUE and FALSE). The target assembler information defines memory addressing formats (scratch pad and array item access), pseudo-operations (data definitions, memory allocation and reservation, library referencing), and special assembler characters available for RTMPL-created-names. Simulator logical states can be defined to allow the user to access simulator firmware (flags and latches), and processor status register information (overflow, result positive, negative, etc.). Because the RTMPL utility produces assembly language source files that are made up of general macro statements, the target definition file must specify which macros are available for the translation process. For example, should an integer addition operation be required, the utility will consult the appropriate target definition file to determine which precisions and argument sources are available for use in implementing the operation. It will then select the add macro that is best suited for time-efficient execution and insert the associated housekeeping operations (load, store, precision conversion) as required. Similarly, if the utility detects the need for data transfer the target definition file is consulted to determine if the appropriate transfer path is supported. This translation approach allows the user to conveniently target a simulation and be assured of obtaining valid time-efficient code (limited by the efficiency of the utility). However, this approach requires that the user be familiar not only with the RTMPL language capabilities but with the specific features of the target simulator.

It is expected that target definition files will normally be formulated by system programmers. Any number of targets may be defined. Target definition is a two-step process (1) writing time optimized assembly language macros to support the required RTMPL features, and (2) defining the characteristics of the macros, using records that are formatted according to RTMPL specifications. Step 1 requires a talented assembly language programmer. Step 2 is accomplished using the SYSDEF utility (fig. 2). This LeRC-generated utility allows the system programmer to build, edit, and list these records in response to prompts from SYSDEF.

The RTMPL targeting capability offers a number of advantages. First, it makes the RTMPL utility

independent of the target hardware. Thus, simulations can be ported from simulator to simulator with minimal effort. Second, it allows for future expansion of RTMPL. Unary and multivariable functions (e.g , trig functions, table lookups, etc.) can be developed as inline macros and they will be automatically included in the RTMPL system. A library of procedures can also be defined in the target definition files, permitting these procedures to be referenced in RTMPL source programs. Finally, the targeting approach permits full exploitation of a target processor's capabilities. RTMPL supports the boolean data type and three arithmetic data types (Integer, Scaled Fraction, and Floating Point), three arithmetic precisions (single, double, and triple) and three operand sources (register, memory, and immediate data). For example, it is possible to define up to 81 addition macros thereby ensuring that the RTMPL utility will be able to select the one most appropriate during parsing.

The RTMPL utility is, in effect, an assembly language programmer. A register assignment algorithm is used to minimize loads and stores. Scratch pad memory usage is automatically invoked when sufficient registers are not available. Precision conversions are also automated, as is scale factor management, when required. All data transfers (both asynchronous and synchronous) are automated including the generation of transfer maps and currency testing at the destination. These features raise the development of multiprocessor simulations to the engineering level.

RTMPL Features

RTMPL is a structured, high-order language which is designed to facilitate the development of error-free, time-efficient simulations. RTMPL source is written by the user in four segments control, global data, local data, and execution. As shown in Fig. 3, separate source files are used to store the control and global data segments. For each processor to be used in the simulation, a single source file (program) is used to specify the local data and execution segments.

Control. In the control segment, the user identifies all of the other source files in the simulation and any output files. The user can also specify options to govern the listing functions of the utility. One somewhat unique option is DEBUG. This option expands the listing to include all major parsing functions and their results. It is particularly valuable during installation of the utility.

Global Data. In the global data segment, the user specifies the data that can be referenced globally by any processor program. The data can consist of messages and/or constants. Messages are used to advise the user of programmed conditions being met during the simulation execution. The user can initiate these advisories using the RTMPL ADVISE command. Constants may be specified by the user to be parametric or fixed. Parameters may be changed at run-time. Changing a global parameter causes it to be changed in all programs in which it is referenced.

Local Data. The local data can consist of three segments constants, variables and argument groups. A constant may be specified to be a simple constant, a one-dimensional array of constants (data table), or a parameter. Only a parameteric constant may be changed by the user at run-time. Variables may be specified to have any number of past values. Past value retention is handled by the utility on the basis of this specification. A variable may be assigned a transfer path (either the real-time or interactive information bus) to satisfy external references. If the transfer path is not specified the utility selects the real-time information bus. Argument groups are used to pass arguments to/from target library procedures. They are also used extensively by the operating system. Argument groups consist of defined variables and/or constants. The user-specified size of the argument group determines the minimum number of items to be contained within the group. The user may 'fill' the group with any number of items, up to the maximum, either using RTMPL or using RTMPOS at run-time.

As appropriate, the user must specify data type, precision, scale-factor and values for all constants and variables. Except for values, the RTMPL utility uses this information to insure data-type compatibility and to recognize the need for housekeeping operations when generating the assembly language code. Values are used by the RTMPOS for its various run-time functions.

Execution. The execution segments contain statements that define the computations to be performed in the simulation. The execution segment is illustrated in Fig. 4. It consists of executives and tasks. Executives contain the main simulation functions. Two types of executives are allowed background and foreground. Any number of background executives can exist, but at least one is required in each execution segment of each processor. The background executive is the main computational program of the processor. Any of the background executives may be selected at run-time, through the RTMPOS, for execution. Up to eight foreground executives are permitted in each execution segment. Foreground executives allow for programmed changes in execution, in a priority interrupt environment, based upon decisions made in the alternate processor in the local channel. That is, the preprocessor can be programmed to interrupt the background executive and initiate a foreground executive in the computational processor and vice versa. The execution of foreground executives is accomplished using the RTMPL ACTIVATE command.

Tasks are used to partition the executable segment into parts which have some operational significance. Tasks are re-entrant and may be entered from any executive in the execution segment using the RTMPL ENTER command. Tasks may be enabled or disabled in executives or at run-time using RTMPOS. The RTMPL DISPATCH command allows the user to force a data-flow type of task execution. Any task, referenced by the DISPATCH command, will be executed when its external variable arguments become available from computations on another processor. These tasks are executed on a first-ready, first-serve basis and dispatching continues until all referenced tasks have been executed.

Tasks and executives are written in terms of statements. There are three types of RTMPL

statements

```
equivalence
<VARIABLE > = <EXPRESSION >
conditional
IF <BOOLEAN CONDITION >
  THEN .. ELSE ..
command
<RTMPL COMMAND > <COMMAND ARGUMENT >
```

The equivalence statement requires data-type compatibility across the equal sign. In the conditional statement the boolean condition may be a boolean expression or a comparison of arithmetic expressions, or a simulator logical state. The 'then' and 'else' clauses may consist of one or more statements. Some RTMPL command statements have already been discussed. The complete set will be described in the forthcoming RTMPL User's Manual.

Expressions may contain any operations defined in the target file for the required data-type Operands must be defined locally, globally, or externally (in another processor program). The expanded format for RTMPL operand specification allows the user to reference, as an operand, any variable or constant defined in the simulation by specifying the channel name and processor type in which the variable or constant is defined. External reference of a constant causes the RTMPL utility to create a constant of the same name in the local program. External variable reference causes a transfer map to be formulated for this variable in its source processor program, and insertion of appropriate data transfer macros in the source and destination programs.

Output Listing

The RTMPL utility provides a comprehensive listing that should aid the user in debugging, optimizing and running the simulation. The listing is organized into parts and the extent of the listing is user-selectable. The first part of the listing consists of an error scan of all source programs. It identifies syntax errors with understandable error messages. For example

ERROR UNDEFINED VARIABLE FUELFLOW

advises the user that the variable FUELFLOW has not been defined in the local data segment. Additionally, the scan provides warning messages whenever a precision conversion or scaling adjustment is necessary for a variable or constant.

In the second part of the listing the RTMPL utility interprets the program source files to allow the user to verify the structure of the simulation. Each statement of each executive and task is listed along with its user or utility assigned label. These labels are used to relate the RTMPL source to the assembly language source generated by the utility. Indentation is used to show the interpretation of nested conditional statements. The execution time of each statement, as determined from the target definition files, is listed along with the statement. Following the listing of each executive and task, the total maximum path execution time is listed. Finally, at the end of each program listing, all external variables referenced in the program are identified to indicate possible sources of computational delay due to interactions with other programs

The final part of the listing provides a tabulation of the data-base, formulated for RTMPOS. It contains extensive information, pertinent to the simulation. This information is alphabetized and formatted for convenient reference. Supplementary listings of the resulting assembler source files are available using a VERSAdos utility.

CONCLUDING REMARKS

A real-time multiprocessor Programming Language (RTMPL) has been designed to aid in the development of multiprocessor simulations. The RTMPL utility (translator) is written, for the most part, in PASCAL to enhance the portability of the language and resulting simulations. The utility consists of about 12 000 lines of code divided into 87 subprograms and requiring about 180 K bytes of resident memory. The utility output is targetable allowing for transportation of simulations between simulators.

The current version of the utility, running on the Motorola EXORmacs system, requires about 20 minutes to translate and list a relatively simple 3-channel (6-processor) simulation. The long translation time results from the multitude of disk file accesses required by its targeting mechanisms. A comparison of projected MC68000 execution times using RTMPL generated code and a hand-coded version of the assembly program showed that RTMPL is nearly 99 percent efficient assembly language programmer.

RTMPL is presently being used at LeRC to program benchmark engine simulations for running on the experimental RTMPS system. It is expected that the current utility design will be frozen for the duration of those tests. Experience gained using this version will suggest design enhancements for inclusion in a future version of RTMPL.

REFERENCE

1. Cole, Gary L. Operating System for a Real-Time Multiprocessor Propulsion System Simulator. Presented at the Summer Computer Simulation Conference, Boston, MA, July 23-25, 1984. NASA TM.

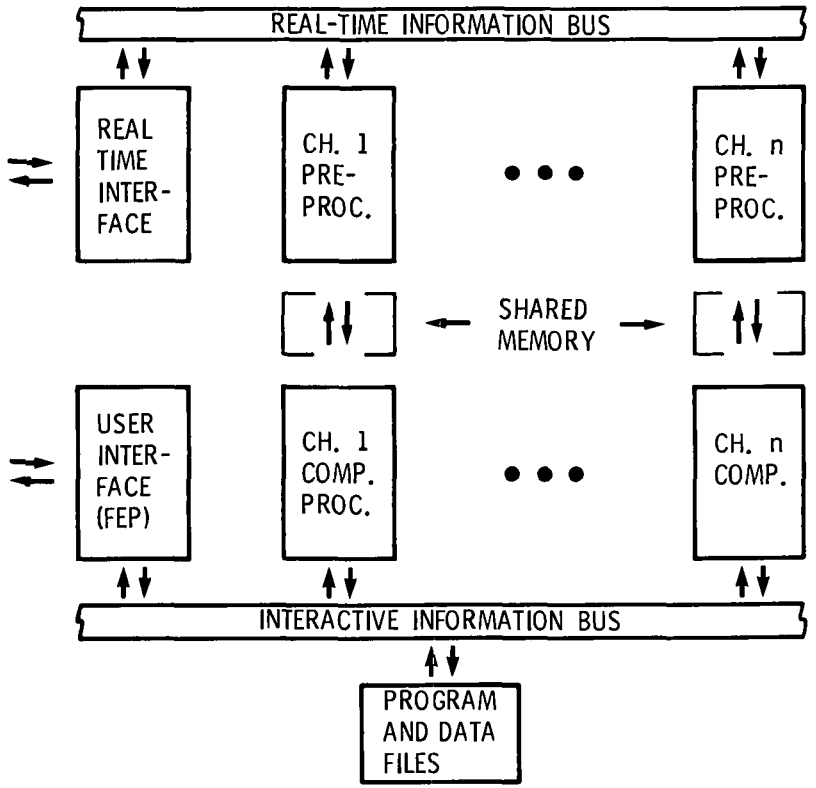


Figure 1. - General simulator configuration.

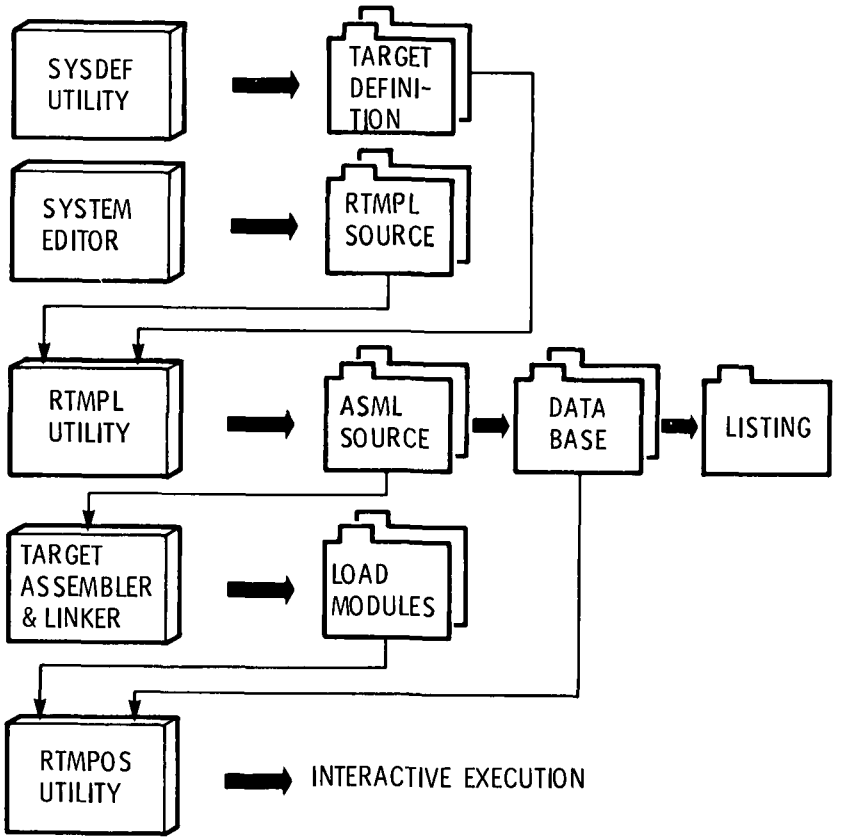


Figure 2. - RTMPS software utilities.

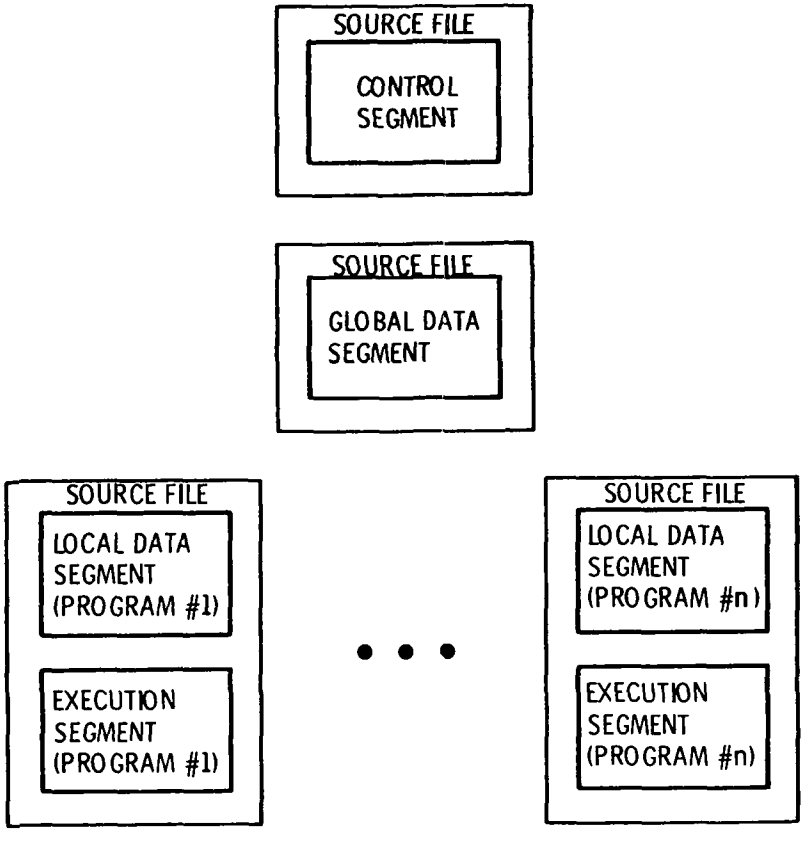


Figure 3. - RTMPL source files.

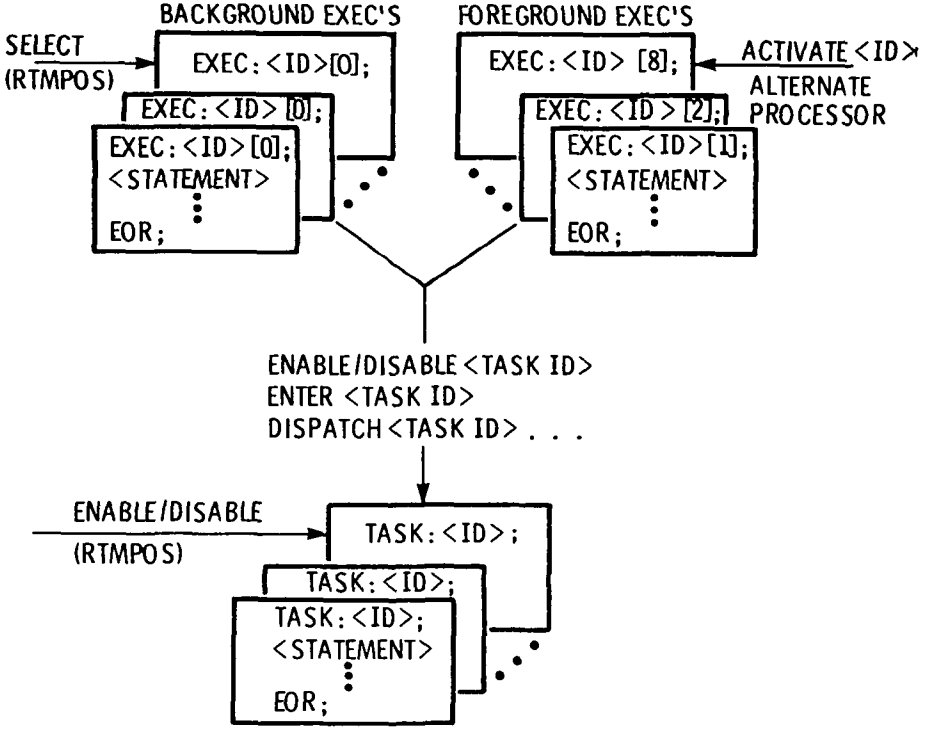


Figure 4. - RTMPL source records - execution segment.

| 1 Report No | 2 Government Accession No | 3 Recipient's Catalog No |
|---|---|---|
| NASA TM-83606 | | |

| 4 Title and Subtitle | 5 Report Date |
|---|---|
| RTMPL - A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulations | 6 Performing Organization Code: 505-40-5B |

| 7 Author(s) | 8 Performing Organization Report No |
|---|---|
| Dale J. Arpasi | E-2025 |
| | 10 Work Unit No |

| 9 Performing Organization Name and Address | 11 Contract or Grant No |
|---|---|
| National Aeronautics and Space Administration<br>Lewis Research Center<br>Cleveland, Ohio 44135 | 13 Type of Report and Period Covered: Technical Memorandum |
| 12 Sponsoring Agency Name and Address | 14 Sponsoring Agency Code |
| National Aeronautics and Space Administration<br>Washington, D.C. 20546 | |

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 simulations of air-breathing propulsion systems. The Real-Time Multiprocessor Simulator (RTMPS) project is aimed at developing a prototype simulator system that uses multiple microprocessors to achieve the desired computing speed and accuracy at relatively low cost. Software utilities are being developed to provide engineering-level programming and interactive operation of the simulator. Two major software development efforts were undertaken in the RTMPS project. A real-time multiprocessor operating system (subject of a companion paper) was developed to provide for interactive operation of the simulator. The second effort (the subject of this paper) was aimed at developing a structured, high-level, engineering-oriented programming language and translator that would facilitate the programming of the simulator. The Real-Time Multiprocessor Programming Language (RTMPL) allows the user to describe simulation tasks for each processor in a straight-forward, structured manner. The RTMPL utility acts as an assembly language programmer, translating the high-level simulation description into time-efficient assembly language code for the processors. The utility sets up all of the interfaces between the simulator hardware, firmware, and operating system. All required interprocessor communications (i.e. data transfers) are automatically established by the utility. RTMPL simulations are self-documenting since the utility produces listings, error messages, warnings, and database files that can aid in the debugging and running of a simulation. The RTMPL utility is written in Pascal. The current implementation runs on a Motorola EXORmacs Development System (host for experimental simulator) and produces MC68000 assembly language code. The utility can easily be retargeted for different processors by changing the contents of a target definition file. The RTMPL language (i.e. command and operation set) is macro-based and can be modified/expanded to meet particular simulation needs.

| 17. Key Words (Suggested by Author(s)) | 18. Distribution Statement |
|---|---|
| Program generators<br>Multiprocessors<br>Real-time simulation | Unclassified - unlimited<br>STAR Category 62 |

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

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

National Aeronautics and
Space Administration

Washington, D.C.
20546

Official Business
Penalty for Private Use, $300

SPECIAL FOURTH CLASS MAIL
BOOK


Postage and Fees Paid
National Aeronautics and
Space Administration
NASA-451

NASA

POSTMASTER: If Undeliverable (Section 158
Postal Manual) Do Not Return