AFIT/GCS/ENG/92D-02

(1)

A MODEL FOR DETERMINING TASK SET SCHEDULABILITY IN THE PRESENCE OF SYSTEM EFFECTS

THESIS

Rusty Olen Baldwin
Captain, USAF

AFIT/GCS/ENG/92D-02

Approved for public release; distribution unlimited

AFIT/GCS/ENG/92D-02

A MODEL FOR DETERMINING TASK SET SCHEDULABILITY IN THE PRESENCE OF SYSTEM EFFECTS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science (Computer Engineering)

Rusty Olen Baldwin, B.S.E.E.

Captain, USAF

December, 1992

Approved for public release; distribution unlimited

Acknowledgements

Research does not occur in a vacuum (well, physics research sometimes does I suppose), it requires the support and involvement of many people. Thanks go to Donna Morris, Clive Benjamin, and Otheos Jackson of Wright Lab for providing the equipment necessary to conduct this research.

I would also like to express my deep respect for and gratitude to Major Paul Bailor. His support, technical expertise, and the high standards of excellence he required provided the motivation and encouragement needed for an effort such as this to be successful.

To my wife and son, Heather and Ian, I would like to thank you, especially, for putting up with my long nights and frequent absences, and also for your loving encouragement. Finally to the most recent member of the family, Nathan, I would like to thank you for delaying your arrival long enough to allow me to complete this thesis. Now let's get Ian and go play!

Rusty Olen Baldwin

Table of Contents

List of Figures

List of Tables

Abstract

This research developed a parameterized model that accounts for system overhead and determines when an Ada runtime environment can no longer successfully execute a given Ada task set and still meet all deadlines. The Ada Compiler Evaluation Capability benchmark was used to characterize an actual runtime environment. Using that data, a generic model of a preemptive, rate monotonic priority based runtime system was developed which accounts for overhead due to clock updates, context switching, task suspension, and synchronization. Validation was based on the Hartstone benchmark. First, the benchmark was executed using the actual runtime environment. Then, those results were compared with the execution of the benchmark using the model. In all cases, except one, the model predicted the point where the task set would fail. A runtime system optimization omitted from model caused the single failure. Experiments conducted using the model allowed the demonstration of the following results. System overhead can be modeled within the existing framework of rate monotonic scheduling theory. Runtime optimizations can be extremely sensitive to phase relationships between task periods and workloads and can render a schedulable task set unschedulable. Requirements of the task set and the performance of the runtime system must be considered simultaneously.

A MODEL FOR DETERMINING TASK SET SCHEDULABILITY IN THE PRESENCE OF SYSTEM EFFECTS

I. Introduction

1.1 Background

An embedded system is a computer system whose main purpose is other than computational, and in many cases, it is used to react to stimuli from its environment "rapidly enough" to control that environment (2). Embedded systems can range from a single microprocessor to a network of large computers and are found in a wide variety of areas. Typical applications of embedded systems include flight control systems in aircraft and missiles, chemical process controllers, control applications in nuclear reactors, data acquisition systems, and environmental control systems. The key phrase in the above definition is rapidly enough. In the context of this research, rapidly enough means real-time – more specifically hard real-time. In a hard real-time embedded system, if the system does not react rapidly enough and it misses a deadline, a catastrophic failure will occur. A catastrophic failure may result in the loss of life, property, irrecoverable loss of data, or a combination of the three. Therefore, the programs which run on embedded computer systems must not only be functionally correct, they must be temporally correct as well.

Given that the timing correctness of a hard real-time system is vital, how should the processor(s) in an embedded system be utilized such that all tasks that the processor must execute will execute rapidly enough to affect the environment? The area of research that investigates this problem is known as scheduling theory. The goal of any scheduling theory is to determine how to schedule a set of tasks with deadlines on a processor or processors so that all the deadlines are met. This has been widely studied on a theoretical level. Often, however, there is a discrepancy between what the scheduling theory predicts and what is actually observed when the embedded system

is built (16:i). The processing reserve capacity is frequently much less than analysis indicated it should have been and/or task deadlines are missed when analysis indicated they would be met. One cause of this discrepancy can be attributed to system overhead and blocking not accounted for by the scheduling theory. Some examples of system overhead and blocking include context switching time, task rendezvous (or synchronization), I/O blocking time, shared data access, and garbage collection. Not accounting for these types of system overhead within the scheduling theory can prove to be enormously expensive to correct in a system design. These types of discrepancies were encountered when estimating throughput requirements for the Navy F/A-18A and A-12 aircraft programs (16:33). By regulation, the Navy required a 50% throughput reserve. Based on the estimation techniques used by the system designers, the Naval Avionics Center (NAC) determined that the actual throughput reserve was significantly less than that. The Navy noted that correcting these deficiencies in an existing design "... is technically challenging, and can add months to a schedule, as well as depleting large amounts of money from the program budget" (16:33).

In contrast to overutilizing a processor's capacity, another possibility (although far less likely) is that not enough of the processor's capacity was utilized. The embedded system could have been built with a less powerful (less expensive) processor. Whatever the outcome, time, resources, and money have been wasted. In addition, the potential for the ultimate failure of the design has been introduced into the system. It is essential, then, that the scheduling theory used to design the embedded system be accurate.

1.2 Problem and Research Objectives

Requirements specifications for embedded systems often use CPU reserve capacity as a design parameter (16, 23). The reserve capacity is then used as an evaluation criteria when determining whether the final design meets specifications. In addition, the reserve capacity of a processor is a fundamental limitation on how much work a given design can perform and hence, a fundamental

limitation on the expandability of the design. It is important, then, that the reserve capacity of a processor be estimated accurately during the design phase before the system is built.

Reserve capacity is defined as the amount of additional processing time available after the processor has completed a pre-defined amount of work. It is expressed as a percentage of a pre-defined total execution time over which the pre-defined workload is distributed. A processor with eight units of processing to complete in ten units of time has a reserve capacity of 20%. The eight units of processing time includes both task execution time as well as any system overhead incurred as a result of the task execution. It is important to note that not all of the 20% reserve capacity will be available for task execution. A portion of it will be consumed by system overhead.

System overhead can be informally defined as the cost of managing system resources necessary to execute user tasks. If the cost of managing those system resources is too great, user tasks can miss their deadlines, even when sufficient processing capacity exists. Therefore, reasonable assurance is needed, early in a system's design that: (1) sufficient reserve capacity is maintained as dictated by the design, and (2) that the system overhead is not so costly that it causes user tasks to miss deadlines.

Rate monotonic analysis (RMA) has been proven to be a significant benefit in both the area of predicting reserve capacity and predictable task scheduling (22). Rate monotonic scheduling theory was first introduced by Liu and Layland (11). The name of the scheduling theory reflects the strategy used to schedule processes or tasks. Tasks are given execution priorities based solely on their rate (how often they execute). The higher the rate of the task, the higher the priority (i.e. a monotonically increasing function of the rate). Although to a lesser degree than many scheduling theories, the rate monotonic algorithm also suffers from a discrepancy between what is predicted to occur and what is observed on a real machine where system overhead is an unavoidable factor. This discrepancy leads to the following research objectives.

1) To demonstrate that intimate knowledge of the entire runtime environment is not required to make an accurate determination of reserve capacity and schedulability – that is, a subset of key runtime parameters is sufficient.

2) To provide insight into an application task's interaction with the runtime environment during execution.

3) To develop a parameterized model of a runtime environment which will provide a conservative determination of task schedulability and processor reserve capacity.

1.2.1 Research Hypothesis The primary hypothesis of this research is that any system overhead (system tasks executed at other than rate monotonic priorities) has the same effect on task schedulability as a lower priority application task blocking the execution of a higher priority user task. A natural result of this hypothesis, if true, is that system overhead, sufficiently characterized and accurately measured, can be modeled in the same manner as user task priority inversion or non-preemptable sections of code. The ability to accurately predict the effect of system overhead can significantly reduce the risk associated with estimating processor capacity requirements and determining task schedulability early in the design of a system.

1.2.2 Scope This research effort was limited to a uniprocessor executing two classes of task sets: independent periodic tasks and dependent periodic tasks. Independent means that the tasks do not communicate or share resources (other than the CPU). Dependent means that tasks must synchronize or communicate at one or more points during their execution.

1.3 Approach

A different way to state the problem this research investigated is: how does the execution of system tasks (or overhead) that do not follow the priority assignments of the RMA affect the

reserve capacity of the processor (clearly it decreases it, but by how much?). Further, how do these same system tasks affect the schedulability of the user task set.

The general approach used to investigate this was: (1) predict the behavior of a given task set by modeling the runtime system tasks and the user task sets interaction with them, (2) download an executable image of that user task set to the target processor and observe the actual behavior. The results of (1) and (2) were compared to refine the model in order to increase its accuracy. The task set executed on the target processor served to validate the model.

The behavior of a given task set was determined by simulating the services a runtime system provides and accounting for the CPU time those services require. For example, if Task A requests synchronization with Task B, the runtime system will require access to the CPU in order to determine whether Task B is ready to synchronize. If Task B is ready, the runtime system will perform the synchronization. If Task B is not ready, the runtime system will place Task A on a queue to wait for Task B. The amount of time that the runtime system requires the CPU will directly affect whether user task deadlines will be met. More specific details of the methodology and the runtime system model are found in Chapters III and IV.

The tools needed to support this approach include: a higher order language in which the task set is written, target hardware, a method of identifying and measuring system tasks in order to construct a model of them, and a task set to execute.

1.3.1 High Order Language Ada was chosen as the Higher Order Language (HOL) for the following reasons: (1) Ada is the official HOL of the DoD and therefore research related to its use in embedded systems is of benefit to programs using Ada, and (2) Ada was specifically designed for embedded systems and contains language constructs which allow for investigation of a task set's behavior at the HOL level. While Ada was used in this research, the results apply to any HOL in which the execution of user tasks follow the RMA priority assignment scheme.

1.3.2 Target Hardware The target hardware was chosen based on two criteria: (1) the processor should be representative of the type of processor used in embedded systems such as robotics and other control applications, and (2) an Ada cross-compiler must exist and be readily available for the target processor. Based on these two criteria, the Motorola 68020 processor was chosen as the target processor (14:1-5).

1.3.3 System Tasks System tasks are often provided by the compilation system in the form of a runtime environment. In some cases they are developed in conjunction with the application tasks. In either case, to determine the effect of the environment on the user task set, the system functions must be identified and measured. The Ada Compilation Evaluation Capability (ACEC) developed for the Avionics Directorate (WL/AAAF) of Wright Laboratory at Wright-Patterson Air Force Base, Ohio, was used to identify and measure these functions. Specifically, the following items were measured:

1) context switching time
2) DURATION accuracy
3) task synchronization
4) CLOCK evaluation
5) TIME and DURATION evaluation
6) DELAY function and
7) interrupts

1.3.4 Task Set The driving criteria in choosing a task set for this research was that the work performed by a given task should be accurately characterized, easily measurable, and repeatable. Further, the amount of work performed by the task set should be representative of that executed by real-time embedded applications. The *amount* of work performed is of primary importance due

to the objective of the research. That is, the purpose of the task set is not to determine how many MIPS or MFLOPS a processor can execute, but rather to ensure a given amount of work was performed.

To generate this task set, the Hartstone benchmark developed by the Software Engineering Institute (SEI) was used. Three classes of task sets have been defined: (1) periodic (harmonic), (2) periodic (non-harmonic), and (3) periodic (harmonic) with synchronization. Periodic tasks are some of the most common tasks in a real-time system (1:5). The harmonic frequency (task frequencies that are integer multiples of each other) was chosen since that will result in a high theoretic utilization for the RMA. Non-harmonic frequencies were chosen because they have a low theoretic utilization. Each of the classes of task sets will be observed while varying various parameters of the task set: work performed, frequency of execution, and synchronization. A block diagram of the test bed is shown in Figure 1.1 and a description of each component in the test bed can be found in Section 3.3, Page 3-7.

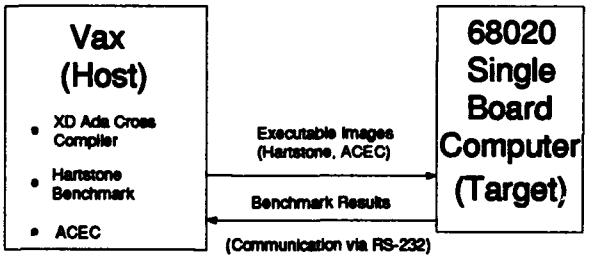

Figure 1.1. Test Bed Block Diagram

1.4 Assumptions

Assumptions about various aspects of this research include:

1) The timing characteristics of the runtime environment (context switch, rendezvous, etc.) will be accurately determined through the ACEC test suite.

2) The Hartstone benchmark reserve capacity and measurement of the processor capacity in Whetstones is accurate.

3) The pipeline architecture of the MC68020 will not introduce any significant error into the timing measurements. A hardware pipeline can increase execution speed by overlapping the execution of instructions at the microcode level. Whether this optimization can occur, however, is highly dependent state of the pipeline (e.g. the particular set of macro instructions in the pipeline) and therefore may not occur consistently in every case. The assumption being made is that any error this may introduce into the timing measurements is so small that it will not materially affect the accuracy of those timing measurements.

1.5 Organization

The remainder of the document is organized in the following manner:

1) Chapter II contains a review of literature used in the course of this research.

2) Chapter III presents a more detailed description of the research methodology used and the test cases used to validate the runtime system model.

3) Chapter IV contains a description of the requirements, design, and functional testing of the model.

4) Chapter V presents the results of the validation tests.

5) The results, conclusions, and recommendations of this research are contained in Chapter VI.

6) Appendix A documents the development and analysis of the equation used to model the XD Ada runtime system implementation of the Ada delay statement. In addition, it contains the raw data used to develop the equation.

7) Appendix B presents the analysis of the XD Ada clock update function.

8) Appendix C contains the raw validation data.

9) Appendix D contains the source code for the model and is available upon request.

10) Appendix E contains the raw ACEC data and is available upon request.

11) Appendix F is a user's manual for the model and is available upon request.

II. Review of Current Literature

2.1 Organization

This chapter is organized into two sections. The first section contains literature dealing with hard real-time scheduling algorithms, especially RMA and extensions to RMA. The second section covers literature dealing with the performance measurement of hard real-time systems.

2.2 Scope

Much of scheduling theory deals with applications in which a statistically fast response is acceptable. These scheduling theories strive to ensure that no task within a system is long deprived of the resource it is requesting. Personal computers, mainframe computers, communications networks, and virtually any multi-user system fall into this category. Scheduling theories in (hard) real-time systems, in contrast, will strive to meet deadlines set by the design even at the cost of never executing lower priority tasks. This review will be limited to literature that addresses scheduling theories used in real-time systems and the performance measurement of real-time systems.

2.3 Overview of Common Scheduling Theories

In any computer system, the scheduling theory (or scheduling algorithm) determines when and by whom system resources are utilized. In a real-time system, an additional constraint of a deadline is added. The scheduling theory in a real-time system not only determines when and who gets system resources but additionally, the "when and who" is subject to higher priority tasks not missing their deadlines. Several methods have been developed to solve the problem of allocating system resources in this manner and they include the following strategies (12):

1) shortest-process-time first

2) earliest-deadline first

3) shortest-slack-time first

4) cyclic executive

5) fixed priority

2.3.1 Shortest-process-time first As the name suggests, tasks that require the least amount of CPU time are given priority in this scheduling strategy. A cursory analysis of shortest-process-time first reveals a characteristic that precludes this scheduling strategy's use in a hard real-time system. The shortest-process-time first does not take into account any deadlines associated with the task. Consider the periodic tasks shown in Table 2.1:

Table 2.1. Shortest-process-time first task set

Tasks	Process Time	Period	Deadline
Task A	1	10	end of period
Task B	3	6	end of period
Task C	4	7	end of period

Assuming the worst case phasing and ignoring system overhead, Task A will run first and complete execution at $T = 1$ before its deadline, Task B will then run completing its execution at $T = 4$ before its deadline, Task C would then run but not complete execution until $T = 8$, one time unit past its deadline.

2.3.2 Earliest-deadline first Earliest-deadline first assigns priority to the task with the closest deadline at any given point in time. The earliest-deadline-first algorithm is optimal in the sense that if a successful schedule for a set of tasks is possible, this algorithm will produce one (11). The drawback to this algorithm, in a real-time environment is twofold: (1) it is computationally expensive to determine whether a set of tasks can be scheduled at an arbitrary instant in time, and (2) if the processor utilization is greater than 100% (i.e. the processor is in an overload condition) the algorithm will fail unpredictably, allowing a task to execute even though it has no chance of meeting its deadline (12). In contrast with shortest-process-time first scheduling which did not account for deadlines, shortest-deadline does not account for processing time.

2.3.3 Shortest-slack-time first Shortest-slack-time first executes the task which has the minimum difference between its deadline and the processing time remaining. In the same manner as earliest-deadline first, shortest-slacktime first is computationally expensive. It also has the effect, in practice, of delaying a task's execution until any preemption at all will cause the task to miss its deadline. Therefore, this algorithm is seldom used (12).

2.3.4 Cyclic executive The cyclic executive model is the traditional scheduling solution of many, if not most, hard real-time systems (22). With a cyclic executive, task executions are explicitly interleaved such that the deadlines of each task can be guaranteed. The schedule is laid out prior to execution so the computational expense of determining a schedule is minimal. A key advantage to the cyclic executive is the predictability of the execution times. As long as task execution times are bounded, task deadlines are guaranteed to be met.

A cyclic schedule is created in the following manner (3). First, the schedule is divided into a fixed time interval called a major cycle (see Figure 2.1). This length of the major cycle must be the least common multiple of the task periods in order to ensure the proper periodicity of the tasks. Each major cycle is divided into frames or minor cycles. Minor cycle boundaries correspond to points where the proper timing is enforced through a timer interrupt. Due to this timing enforcement, the minor cycle can be no longer than shortest period of the process being scheduled. If a task requires an amount of processing time that is greater than one frame, the processing time must be broken into several subactions or chunks and distributed across several frames. Unless a mode change occurs (such as system shutdown) the major cycle is continously repeated.

The cyclic executive is a fragile algorithm in two important aspects. If a task requires more frames to complete its execution than it has been allotted by the schedule, a frame overrun is said to have occurred. Two actions typically take place when a frame overrun occurs; either the task is aborted or an empty frame within the major cycle can be used to handle the extra execution. Of course, depending on the application, either solution may not meet the timing constraints (3).

The fragility is also manifest when designing a cyclic executive schedule. Consider the following example (22). A task set has periods of $100 : 150 : 350 = 2 : 3 : 7$. A minor cycle of 50 would require a major cycle of 42 minor cycles. Any change in the design, periods of the tasks, or CPU time required by the task would require the schedule be reaccomplished. This entails a significant effort, especially if it occurs late in the design phase. Another problem is that much of the processor capacity can remain unused if the task's worst case execution time is much greater than its typical case execution time.

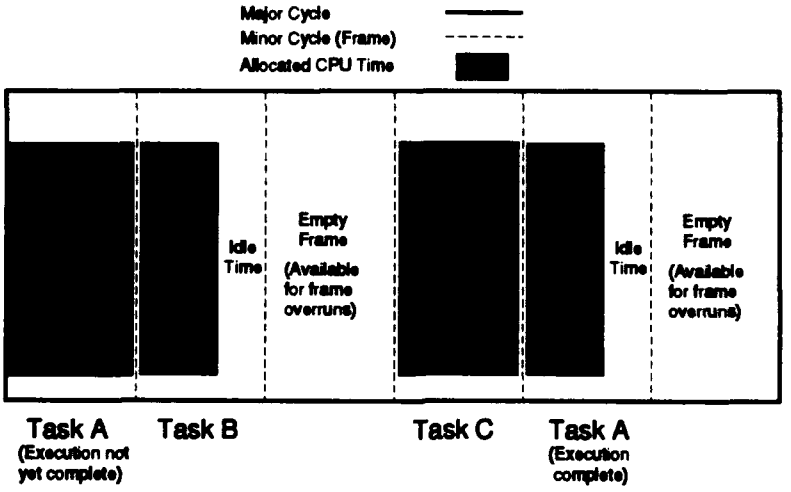

Figure 2.1. Cyclic Executive Schedule

2.3.5 Rate Monotonic A fixed priority algorithm executes tasks based on a priority determined prior to execution. The RMA is a variant of the fixed priority scheduler. In the seminal paper on RMA (11) it was shown that a schedule will always exist for task set with priorities assigned according to the RMA on a processor whose utilization is below a certain bound.

Tasks are assigned static priorities based solely on their rate (how often they execute). The higher the rate of the task, the higher the priority (i.e. a monotonically increasing function of the rate). Essentially, Liu and Layland established that if the sum of the ratios of the time to execute a set of tasks and the periods of the tasks is less than or equal to $n(2^{1/n} - 1)$, (0.693 as the number of

tasks approaches ∞) then those tasks can be scheduled using rate monotonic priority assignment. This relationship is expressed by the following equation.

$$\Sigma \frac{C_i}{T_i} \leq U(n) = n(2^{1/n} - 1)$$

where n is the number of tasks, C_i is the time to execute task i and T_i is the period of task i.

This algorithm was a landmark achievement since it separated the functional correctness of a task from its timing characteristics. As long as the sum of the set of task's ratios is less than or equal to $U(n)$ the task set can be scheduled without regard to any other factor. In fact, this property has the added benefit that even during processor utilization greater than $U(n)$ (even if $U(n)$ is greater than 1), the set of tasks whose ratio is less than $U(n)$ (which is called the stable set) is guaranteed to meet their deadlines. Thus, the RMA is a stable algorithm during overload conditions for the tasks in the stable set.

The theory, however, is limited by assumptions that the authors made. First, the tasks are assumed to be periodic or executed on a regular basis (e.g. every 5 milli-seconds). Second, task communication was not accounted for. Finally, the tasks' execution times are assumed to be bounded by a constant. These assumptions have been relaxed by subsequent research.

2.3.6 Extensions to RMA For all its advantages, the RMA processor utilization bound of 0.693 for guaranteed schedulability is frequently limiting in practice. Lehoczky, et al. extended that bound to 0.88 for a randomly chosen (or average case) set of tasks (9). A theorem was derived to determine exactly if a given task set could be scheduled. The average case behavior of a task set is theoretically interesting because it showed that the rate monotonic theory is applicable to a wider portion of realistic problems.

The average case described, however, suffers from a slow convergence. That is, a large task set is needed for the behavior to be exhibited. The exact characterization of a given task set is more applicable to this research. The theorem derived can determine that, even though the utilization

may be greater than the upper bound determined by Liu and Layland, the tasks can nevertheless be scheduled. The theorem checks each and every scheduling point of the task set. If a schedule exists, it will be found by the following theorem (9).

A set of n independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if and only if

$$\forall i, 1 \leq i \leq n, min(\sum_{j=1}^{i} C_j \frac{1}{lT_k} \lceil \frac{lT_k}{T_j} \rceil) \leq 1$$
$$(k,l) \in R_i$$

where C_j and T_j are the execution time and period of task τ_j respectively and $R_i = \{(k,l) \mid 1 \leq k \leq i, l = 1, \ldots, \lfloor \frac{T_i}{T_k} \rfloor\}$.

Note that although this theorem states "scheduled by the rate monotonic algorithm" it will, in fact, determine if any schedule exists for any task set using a fixed priority assignment scheduling algorithm.

2.3.7 Task Synchronization Dependent tasks (or tasks that synchronize) present a problem for RMA. When tasks synchronize, a higher priority task may be required to delay its execution to wait for a lower priority task. This requirement to wait for a lower priority task is contrary to the primary principle of RMA, namely, that a higher priority task will always preempt a lower priority task. Common synchronization protocols include semaphores, monitors, and Ada rendezvous. Use of these synchronization protocols could lead to a high priority task being blocked indefinitely. Consider the following example.

A task set consists of five tasks T_1, T_2, T_3, T_4, T_5 where the priorities ranked from T_5 (highest) to T_1 (lowest). T_1 and T_5 share a common memory location through a semaphore. Suppose T_1 locks a semaphore and is subsequently preempted by T_5. T_5 also needs the semaphore but must wait for T_1 to unlock it. In the mean time, T_1 is subject to preemption from T_2, T_3, and T_4, multiple

times causing the highest priority task, T_5, to wait arbitrarily long to execute. Clearly, this type of situation is unacceptable in a hard real-time environment.

For this reason, the priority ceiling protocol was developed (22). The protocol has two basic tenets. First, if a task blocks the execution of any higher priority task, it will inherit the priority of the highest priority task blocked. Second, a task is only allowed to enter a critical section if the section will always execute at a priority higher than the inherited priority level of any preempted critical sections. This protocol will ensure freedom from mutual deadlock and provide a bounded blocking time. A high priority task using the priority ceiling protocol will be blocked at most once by a lower priority task.

Given these properties of the priority ceiling protocol, the maximum time a higher priority task can be blocked is equivalent to decreasing its deadline by the same amount. If a task's deadline is at $T = 100$ and it can be blocked at most by 30 units of time, its equivalent deadline is now $T = 70$.

To account for this blocking, the following equation applies:

A set of n independent periodic tasks scheduled by the rate monotonic algorithm and using the priority ceiling protocol will always meet its deadlines, for all task phasings, if and only if (21)

$$\forall i, 1 \leq i \leq n, min(\sum_{j=1}^{i-1} C_j \frac{1}{lT_k} \lceil \frac{lT_k}{T_j} \rceil + \frac{C_i}{lT_k} + \frac{B_i}{lT_k}) \leq 1 \qquad (2.1)$$

$$(k, l) \in R_i$$

where C_j and T_j are the execution time and period of task τ_j respectively, $R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \ldots, \lfloor \frac{T_i}{T_k} \rfloor\}$ and B_i is the worst case blocking time for τ_i.

2.4 Performance

This section summarizes the literature dealing with the performance measurement of computer systems. Specifically, this section reviews literature that addressed (1) how to measure the performance of an embedded target, (2) pitfalls associated with performance measurement of computers, and (3) factors that could distort the performance measurement of a given system.

2.4.1 Performance measurement of an Ada compilation system Measuring the performance of a compilation system is a complex procedure (4:760). Three aspects of this measurement are critical for success: isolating the feature to be measured, repeatability, and sufficient accuracy to obtain meaningful results. Typically, isolating the feature to be measured is achieved by executing a control loop without the feature to be measured, and a loop containing the feature to be measured. Theoretically, the time difference between these loops is the time required to execute the feature under test. There are several caveats however. First, if the compilation system clock is being used to measure time, it must be of sufficient precision relative to the feature being measured, or inaccurate results will be obtained. Second, steps should be taken to ensure that any compiler optimization does not interfere with the measurement by optimizing out the very feature that is being measured.

As a case in point, during testing done by the Software Engineering Institute, a certain benchmark test was consistently returning negative time results (26). The cause was eventually determined to be related to the Ada CLOCK resolution of the particular compilation system Factors which may cause variations are (26): clock overhead, optimization, memory allocation, garbage collection, and other operating system effects.

In order to sufficiently characterize the behavior of an Ada compilation system relevant to real-time performance, the following elements should be measured (4:765):

1) subprogram calls

2) task activation

3) task termination

4) task synchronization

5) CLOCK evaluation

6) TIME and DURATION evaluation

7) DELAY function

8) garbage collection and

9) interrupts

Three benchmarks have been generally available to test these important aspects of an Ada compilation system (6):

1) the University of Michigan Ada benchmarks,

2) the Performance Issues Working Group benchmarks (or PIWG, an Association of Computing Machinery special interest group), and

3) the Ada Compiler Evaluation Capability (ACEC).

Over the course of development of these three benchmarks, the functionality of the University of Michigan benchmarks and PIWG benchmarks has been incorporated into the ACEC.

The ACEC Reader's Guide (17) defines what the ACEC is designed to accomplish, the intended users of the package, and the rationale of the design. The ACEC was designed to compare the runtime performance of different compiler implementations as well as to compare various non-performance related aspects of a compiler such as: assessment of a symbolic debugger, library system management, and diagnostic message clarity. The ACEC tests incorporate many features to test for and to defeat many of the compiler optimizations or other effects that would result in

inaccurate results. This test suite is widely used by compiler manufacturers to test their compilers prior to release.

2.4.2 Task set performance measurement A limitation with benchmarks such as the ACEC is that they focus on a single activity or aspect of a compilation system. They do not attempt to measure the performance of a set of activities and whether or not this set of activities (or tasks) meet a given set of performance criteria. In fact, trying to draw general conclusion from such measurements is difficult and risky (1). Since a real-time system will be performing many tasks, though, it is important to establish that the given set of performance criteria is being met.

The Hartstone benchmark is one such benchmark that attempts to quantify whether or not a set of activities meet a given set of performance criteria. The requirements document for the Hartstone (1) defines an operational concept and requirements for a set of benchmarks designed to test the ability of a system to run hard real-time applications. The Hartstone was not designed to test a particular scheduling paradigm or programming language. Although the first implementation of it was in Ada, it was designed to be translated to any language being utilized for hard real-time systems.

The benchmark was designed to have the following characteristics (1).

1) It spans the entire hard real-time problem domain – it contains periodic and aperiodic event driven tasks. The aperiodic events are both user-initiated and interrupt-driven. Task synchronization, access to common data, mode changes and distributed processing are included.

2) The benchmark tests increase in complexity. That is, in the series of tests, simple or (presumably) easy to accomplish tasks are run first followed by tests that are considered more difficult.

3) Each test has a baseline requirement and a strategy for increasing that requirement to stress the system along a number of dimensions. For instance, the periodic workload could be increased as other factors remain constant. Other dimensions that can be independently varied are: processing load, aperiodic events, and task frequency.

4) Each test is self-verifying. The test itself verifies that the computations are being performed correctly and that it met its deadline.

5) The computational load is synthetic. In the case of the Hartstone, a self-verifying version of the Whetstone is used for the computational load. This ensures that when comparing various processors or architectures, the same amount of work is being performed.

6) A relative figure of merit is assigned for each test that clearly distinguishes between actual work being done and system overhead. Therefore, the more useful measure of maximum utilization prior to a deadline being missed can be determined rather than maximum throughput.

The Hartstone consists of five series of tests. Table 2.2 lists the main objective of each test.

Table 2.2. Hartstone Tests

Test Series	Measurement Objective
PH	Periodic Tasks, Harmonic Frequencies
PN	Periodic Tasks, Non-Harmonic Frequencies
AH	Same as PH with APeriodic Tasks Added
SH	Same as PH with Synchronization Added
SA	Same as PH with APeriodic Tasks and Synchronization Added

2.5 Summary and Conclusions

Scheduling theories have received much attention in the literature. Due to the computational expense of most scheduling schemes only the cyclic executive and fixed priority scheduling has seen wide use in the real time applications. A common missing element in most literature, however, is that system overhead issues such as interrupts, task synchronization, and other functions necessary

for the operation of a real system are not addressed in a comprehensive manner. That is, most literature either ignores system effects completely or only addresses a subset of system effects, while ignoring other effects that have an equally pronounced impact on the schedulability of a given task set. The end result is that if a software engineer needs to determine the feasibility of a particular schedule, a model must be constructed from a variety of sources and is likely to be tied to a particular runtime system. A later change in the runtime system might require another model be constructed.

In order to account for system effects, they must be measured. To measure individual features, the standard approach is to isolate the feature to be measured by executing a control loop without the feature, and then a loop containing the feature to be measured. The time difference between these loops is the time required to execute the feature under test. While this method will give useful results, it is not sufficient to enable one to draw any general conclusions about the behavior of the system while running a given task set. For instance, while you may know that a particular runtime environment takes x microseconds to reclaim unused memory (garbage collection), using the control loop approach gives no information about under what conditions unused memory is reclaimed. In fact, when a runtime system does garbage collection may directly affect the schedulability of the user task set. In order to determine the behavior of a system which takes into account system effects that occur during the execution of a task set, other measurement techniques are used. One such approach is to execute a task set that performs a known amount of work and to increase the workload (or another parameter of interest) and observe the effect on the schedulability of the task set.

This research will focus on the construction of a model which will permit a schedulability determination of a given task set under a controlled load. It will, in essence, be a plant/load simulation of a runtime system. Load simulations are widely used as a means of verification of various system properties (10). The contribution this research makes is to show that a parameterized

model of the runtime system (or a parameterized load simulation) can be constructed based on the measurement of a subset of key runtime system features thereby making the model applicable to a wide range of runtime environments. Not only will the model determine whether a particular user task set can be scheduled under a particular runtime system, but also whether it can be scheduled under any runtime system with the same parameters. If, in fact, the runtime system is being designed in concert with the user tasks, the execution budget can be used in the model in place of the actual measurement of particular runtime system. This type of model will permit, early in a system's design, an analysis of whether a particular task set will execute. Additionally, it can also be used to determine the worst case overhead a particular task set can suffer before the task set will no longer meet its deadlines.

III. Methodology

3.1 Introduction

This chapter presents the research methodology used to construct and validate a task scheduling model, RATESIM (RATE monotonic scheduling SIMulation), which predicts the behavior of a given task set. Recall that the objective of this research is to provide a means of determining, early in the design phase, the schedulability of a given user task set while taking into account the effect of the runtime environment (system tasks) in which that user task set must execute. The RATESIM program will be used to fulfill this objective.

First, the system tasks are identified and the methods to measure their effects are presented. Next, user task sets (used for validation) are presented along with the objectives of each validation test. The equipment being used is identified and the test bed configuration is described. Finally, the data to be gathered and the statistical analysis of that data is discussed.

3.2 Research Methodology

In order to meet the objective stated above, the following basic research methodology was used.

1) Identify the system tasks.
2) Measure those system tasks to determine the amount of CPU time they use and at what frequency.
3) Define a user task set.
4) Predict the behavior of the user task sets based on the system tasks and determine the effect the system tasks will have on the schedulability of the given task set (i.e. will the task set still meet all its deadlines?). Specifically, the system tasks will be modeled as a blocking factor that each user task in the task set suffers.

5) Construct and execute the user task set on the target processor and observe whether or not the prediction was accurate.

6) Go to step 3.

The above methodology will serve as the basic framework for investigating the effect system tasks have on the schedulability of a user task set. A discussion of each specific step follows.

3.2.1 Identify the System Tasks System tasks are those functions which are necessary to manage the resources of an embedded system but nevertheless do not directly perform useful work from the perspective of the user task. System tasks *allow or facilitate* the performance of useful work by user tasks. In addition, system tasks often immediately preempt user tasks, without regard to any user deadlines.

The following list is a preliminary set of system tasks which will affect the schedulability of a user task set. These were identified through searching existing literature (4, 26).

1) system CLOCK updates

2) context switching time

3) DELAY resolution

4) interrupts

5) TIME and DURATION evaluation

6) task synchronization

7) task activation

8) task termination

9) garbage collection

While this is certainly not an exhaustive list of items that could affect a user task set when implemented on a real machine, it represents those that have the most pronounced effect (26). Dynamic task creation, termination, and garbage collection are system tasks not included in the RATESIM model.

3.2.2 Measure the System Tasks The Ada Compilation Evaluation Capability (ACEC) test suite (version 3.0) was used to measure the various system tasks which affect the schedulability of the user task set. The ACEC is a compiler evaluation benchmark used to assess the capabilities of Ada compilation systems. Its purpose is: (1) to allow comparison of different compilation systems using objective, measured data and (2) to determine performance characteristics of a given compilation system. The ACEC is intended to measure many aspects of an Ada compilation system including: capacity limits, symbolic debuggers, library management systems, diagnostics, compile time, code size, and execution time. Obviously, this research will limit its use of the ACEC to determine the execution time of system tasks of interest in the compilation system executing on the target hardware.

To measure the above system tasks, runtime performance benchmarks of the ACEC were used coupled with the ACEC Single System Analysis (SSA) tool. The SSA tool extracts information implicit in relationships between various test problems. The SSA major report categories include: Language Feature Overhead, Optimizations, Run-time System Behavior, and Coding Style Variations. Over 1600 tests are included in the ACEC and the SSA can summarize and report the performance of virtually every system task of interest in a variety of execution modes.

Figure 3.1 (18) is an overview of the ACEC. The area surrounded by the dashed line represents the portion of the ACEC not utilized in this research. A detailed description of the ACEC and in depth discussion and analysis of the measurement techniques are beyond the scope of this research. These details can be found in the documents provided with the ACEC (17, 18, 19). A summary of ACEC results are contained in Chapter V.

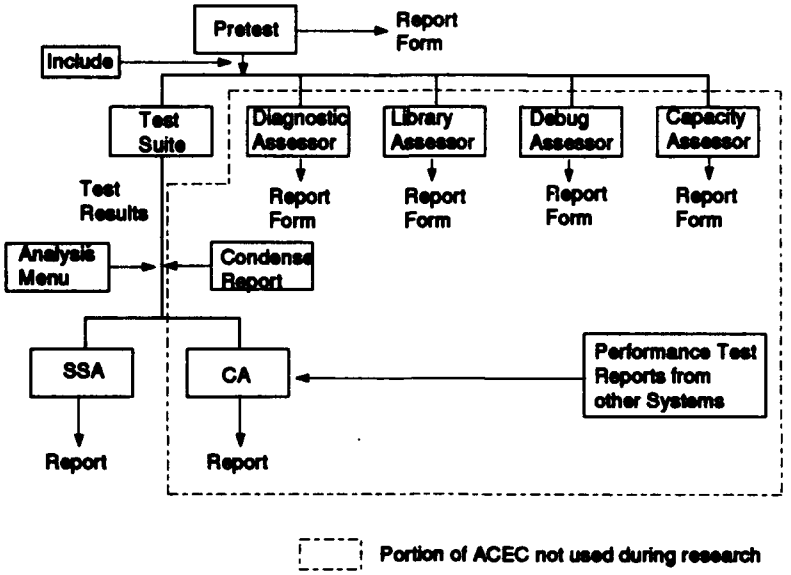

Figure 3.1. Overview of the ACEC

3.2.3 Predict the effect - RATESIM The effect of the system tasks measured by the ACEC on the user task set is determined by incorporating the execution time measurements into the RATESIM model and then submitting the user task set to RATESIM. RATESIM "executes" the user task set (e.g. accounts for the load) and determines, based on the ACEC measurements, the interaction between the user task set and the system tasks.

RATESIM determines whether a user task has met its deadline by accounting for system task and user task utilization of the CPU. Since it knows the deadlines of user tasks it can detect when a user task has missed a deadline.

All system task execution times are parameterized within RATESIM in order to allow for easy modification should one wish to model a different runtime system. In addition, system tasks not specifically parameterized can be easily added by supplying the system task parameters to the model. The most significant behavior that is currently embedded within the RATESIM model (and therefore not easily modified) are: (1) a fixed priority, preemptive, event-driven scheduling strategy, and (2) system tasks will always preempt user tasks. Priorities of user tasks are determined using rate monotonic priority assignment (the higher the rate of the task, the higher the priority). Other

embedded behavior includes two runtime optimizations considered general practice. Details about these optimizations and the RATESIM design is contained in Chapter IV. RATESIM validation details are in Chapter V.

3.2.4 Run the task set The user task sets are based on the SEI Hartstone benchmark version 1.0 (7). This benchmark provides well-defined tasks and allow task set parameters such as workload and frequency to be varied. The benchmark tasks sets have been used as written when appropriate and modified as needed to meet the objectives of this research. For example, the Periodic Tasks, Non-Harmonic Frequencies (PN series) benchmark and the Periodic Tasks, Harmonic Frequencies with Synchronization (SH series) benchmark described in the Hartstone requirements document (1) have not yet been implemented. Therefore, the current Periodic Tasks, Harmonic Frequencies (PH series) benchmark was modified to implement the PN and SH series requirements.

The following task sets have been defined for use to validate the behavior of RATESIM. For each set, the measurement objective of task set has been identified.

3.2.4.1 Task Set A This task set consists of five periodic tasks (see Table 3.1) whose frequencies are integer multiples of each other (harmonic). This task set represents a major class of real-time applications. It has a high theoretical utilization based on RMA. Two parameters of this task set are varied, the work performed (C_j, expressed in kilo-whetstones) and the frequency (T_j). The objective in using this task set is to observe the interaction between the system tasks and C_j. As C_j is increased, the system overhead should initially remain constant since no additional overhead is induced. Conversely, as T_j is increased, user task CPU utilization should decrease immediately due to an increase in task switching overhead.

3.2.4.2 Task Set B This task consists of a set of five periodic tasks (see Table 3.2) whose frequencies are non-harmonic. This task set has a low theoretical utilization based on RMA. The difference between this task set and task set A is that the system overhead should be

Table 3.1. Task Set A - Periodic/Harmonic

Task No.	Frequency (Hertz)	Kilo-Whetstones per Period	Kilo-Whetstones per Second	Requested Workload Utilization
1	2.00	32	64	4.79%
2	4.00	16	64	4.79%
3	8.00	8	64	4.79%
4	16.00	4	64	4.79%
5	32.00	2	64	4.79%

significantly greater to start with due to the non-harmonic frequencies of the tasks. Non-harmonic relationships between tasks will often defeat optimizations that the runtime system could normally implement with harmonic task sets. This behavior is discussed further in Chapter V, Section 5.5.2.

Table 3.2. Task Set B - Periodic/Non-Harmonic

Task No.	Frequency (Hertz)	Kilo-Whetstones per Period	Kilo-Whetstones per Second	Requested Workload Utilization
1	2.00	32	64	4.79%
2	2.30	16	36.8	2.75%
3	4.59	8	36.7	2.75%
4	6.89	4	27.6	2.06%
5	9.19	2	18.4	1.38%

3.2.4.3 Task Set C This task consists of a set of five periodic tasks in two different configurations(see Tables 3.3 and 3.4). The task set frequencies are integer multiples of each other (harmonic) and they have synchronization requirements. The parameter to be varied is the frequency of each task in the task set. The objective of this task set is to validate the behavior of the model during synchronization. Tasks that execute a workload (Task 5 in Task Set C2 does not) have an initial requested utilization of 4.79% per period.

Table 3.3. Task Set C1 - Periodic/Harmonic/Synchronization

Task No.	Frequency (Hertz)	Kilo-Whets per Period	Kilo-Whets per Second	Entry Call time (μs)	Accept at at time (μs)	Accept Workload (KW)
1	2.00	32	64	n/a	n/a	n/a
2	4.00	16	64	n/a	n/a	n/a
3	8.00	8	64	n/a	n/a	n/a
4	32.00	4	64	n/a	0	0
5	32.00	2	64	0	n/a	n/a

Table 3.4. Task Set C2 - Periodic/Harmonic/Synchronization

Task No.	Frequency (Hertz)	Kilo-Whets per Period	Kilo-Whets per Second	Entry Call at time (μs)	Accept at time (μs)	Accept Workload (KW)
1	2.00	32	64	n/a	n/a	n/a
2	4.00	16	64	n/a	n/a	n/a
3	8.00	8	64	n/a	n/a	n/a
4	32.00	4	64	n/a	0	2
5	32.00	0	64	0	n/a	n/a

3.3 Equipment

Table 3.5 lists the equipment being used. A block diagram of the test bed is shown in Figure 3.2.

Table 3.5. Research Equipment and software

Equipment	Comment
Motorola 68020	target processor (the unit under test)
Vaxstation III	development computer for SW benchmarks
XD Ada cross-compiler	Ada compiler for 68020
Hartstone benchmark	test cases
ACEC test suite	measure compiler runtime performance

3.3.1 Target system The target processor is a Motorola 68020 on the MVME133A-20 Monoboard Microcomputer (15). The MVME133A consists of the MC68020 microprocessor and the MC68881 coprocessor both running at a clock speed of 20 MHz. There is 1 MByte of dynamic RAM onboard.

3.3.2 Host system The host system is a Vaxstation III running the VMS 5.1 operating system. Communication to the target is provided through two serial ports. One serial port is used to download the kernel and application code and the other is used as a debug communications port. In this research the debug port was used to display messages from the application code.

3.3.3 Compilation System The compilation system used was the System Designers XD Ada MC68020 cross compilation system (24). XD Ada consists of a cross-compiler, development tools (builder, loader, assembler, librarian, etc.), debugger, predefined compilation units, and a run-time object code library.

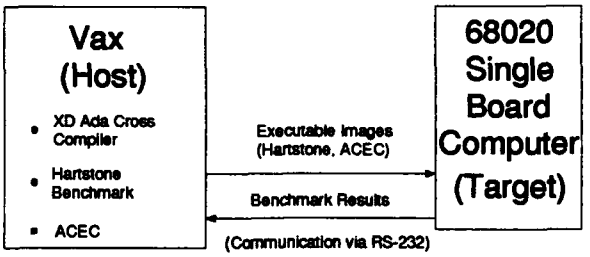

Figure 3.2. Test Bed Block Diagram

3.4 Data

Data collected during this research came from three sources: the Hartstone benchmark, XD Ada runtime kernel, and the ACEC. The data collected from those sources is listed in Table 3.6. Table 3.7 explains the meaning of each data item.

Table 3.6. Research Data

Data Item	Source
Task Set	Hartstone/ User generated
Raw Speed	Hartstone benchmark
Task frequency	Hartstone benchmark
Workload	Hartstone benchmark
Met Deadlines	Hartstone benchmark
Missed Deadlines	Hartstone benchmark
Skipped Deadlines	Hartstone benchmark
Average Late	Hartstone benchmark
Task Utilization	Hartstone benchmark
Context Switch Time	ACEC
Delay Time	ACEC
Rendezvous Time	ACEC
Clock Update Time	XD Ada kernel

Table 3.7. Data Definitions

Data Item	Comment
Raw Speed	Raw CPU speed in Kilo-Whetstones
Task frequency	Number of times per second the task is required to perform the requested workload
Workload	Amount of work required of the task in Kilo-Whetstones
Met Deadlines	Number of times during the test that the task successfully completed its workload before the next scheduled activation time
Missed Deadlines	Number of times during the test that the task failed to complete its workload before the next scheduled activation time
Skipped Deadlines	Number of scheduled activation times which were not performed due to a previously missed deadline
Average Late	Average lateness of missed/skipped deadlines
Task Utilization	Percentage of CPU time dedicated to user tasks
Context Switch Time	Worst case execution time to perform a context switch
Delay Time	Actual Delay time
Rendezvous Time	Worst case execution time to perform a rendezvous
Clock Update Time	Worst case execution time to update system clock

3.5 Data Analysis

The data collected during this research was used in two ways: (1) to describe the runtime environment, and (2) to validate the RATESIM model behavior. The data collected using the ACEC test suite was used to supply execution time of key runtime system functions – this served as the basis for the description of the runtime system. The data collected Hartstone benchmark served as the standard for determining "correct" behavior in the RATESIM model. Data from the Hartstone benchmark and the RATESIM model were compared and any significant differences in the data served to identify deficiencies in RATESIM. The model was then modified to correct those deficiencies and the tests were rerun.

The first task in describing the behavior of the runtime environment was to analyze the data gathered by the ACEC and construct a model of the individual runtime services being modeled by RATESIM. The model was constructed to be able to "execute" independent and dependent, periodic tasks. The following runtime services from the runtime system were needed as input to the model: clock update time, context switch time, rendezvous time, and DELAY expiration time. The following sections summarize the analysis that was performed for each of the services.

3.5.1 Clock Update Time The clock update of an Ada runtime system is the system function used by an Ada program to provide any time-related requests of a program such as: TIME, YEAR, SECONDS, DELAY, etc. The clock update service is typically interrupt-driven, periodic, and very short. Even though it is typically short, it is a runtime function which will be a source of non-rate monotonic utilization of the CPU and therefore could affect the schedulability of user tasks.

Since the clock update involves a relatively small amount of code, it lends itself to manual code analysis to determine execution time. The analysis for the XD Ada runtime system clock update is included in Appendix B and was determined take $15.4\mu s$ to update the clock every $41,600\mu s$.

3.5.2 Context Switch Time Context switch time is the time required to save the state of the task currently executing and restoring the state of the task to switch to. The ACEC provides a measure of the context switch time and determines a confidence interval. RATESIM uses the worst case execution time for the context switch. Further details of the context switch time measurement can be found in Chapter V.

3.5.3 Rendezvous Time Rendezvous time is the time required for two tasks to synchronize their execution at a given point in time. The ACEC provides several measurements of rendezvous time. It measures rendezvous with and without parameters, rendezvous when the calling task arrives first and when the called task arrives first, and many different measurements of various combinations of selects and timed entry calls. RATESIM modeled a simple rendezvous (no parameters, or select alternatives) and used the worst case execution time for the rendezvous without regard to whether the calling task or the accepting task arrived first. Further details of the rendezvous time measurement can be found in Chapter V.

3.5.4 DELAY Expiration Time An Ada DELAY statement introduces a significant amount of variability into the runtime behavior of a task set. The Ada Language Reference Manual (LRM) (5) requires only that the DELAY statement provide "... at least the duration specified ...". Obviously, this type of behavior is unacceptable in a real-time environment. Therefore, any runtime system designed for a real-time application will provide a DELAY statement that is more predictable than the LRM requires. Since the DELAY statement is dependent on the timing of the runtime environment for implementation and the runtime environment is unique to each particular compilation system, the behavior of the DELAY statement is not predictable between various implementations.

The ACEC will measure the difference between the delay requested by a program and the actual delay provided by the runtime system. One approach to modeling the DELAY statement is to simply add the worst case additional delay as determined by the ACEC to the requested delay.

The ACEC measurements of the XD Ada DELAY implementation showed that the additional delay observed could vary by as much as 447μs. This type of variability will result in poor prediction of schedulability of a task set which would otherwise be schedulable.

Based on the ACEC measurements, a model of the DELAY implementation of the XD Ada runtime system was constructed. This model accounted for 143μs of the DELAY statement variability. An additional 149μs can be attributed to context switching leaving 155μs still unaccounted for. An analysis of the DELAY implementation of the XD Ada runtime system and the equation developed to model it can be found in Appendix A.

3.6 Summary

This chapter described the research methodology used to construct and validate the RATESIM model. It consisted of identifying the system tasks to be modeled and measuring those tasks, constructing the user task sets and repeatedly comparing the behavior of a given user task set on an actual machine to the behavior predicted by RATESIM. These comparisons provided the basis for refining the RATESIM model in order to make it more accurate.

The ACEC was the method used to determine the execution times of the identified system tasks. A summary of the ACEC and a summary of the analysis of the collected ACEC data was presented. The system task models contained within RATESIM was a direct result of this analysis.

The user task sets used for validation of RATESIM consisted of three classes of tasks: periodic/harmonic, periodic/non-harmonic, and periodic/harmonic with synchronization. The test bed used during this research consisted of a Vaxstation III host and a MC68020 single board computer serving as the target processor.

IV. Model Requirements, Design, and Testing

This chapter describes the purpose and objectives of the RATESIM model. It documents the requirements which resulted in the final design and presents the environmental and behavioral models of RATESIM. Finally, the test plan and test results to verify the operational behavior of RATESIM is presented.

4.1 Purpose and Objectives

The purpose of RATESIM is to model an embedded runtime system that will "execute" (i.e. account for the CPU utilization of) a given task set and monitor the user task set interaction with the runtime system.

The primary objective of the RATESIM model is to determine whether a user task set can execute on a real processor and its associated runtime system without missing any of its deadlines. This determination is based on (1) the user task set, (2) the execution time of key parameters of the runtime system, and (3) the scheduling strategy of the runtime system. The secondary objective is to provide insight into the user task set interaction with the runtime system. This insight will be provided through an event history of the user task set's interaction with the runtime system and statistics based on that interaction such as CPU time allocated to user tasks and user deadlines missed.

4.2 Motivation

The motivation for building the RATESIM model came early in the research effort. Initially, this research focused on constructing a general mathematical model for the blocking term (B_i) in Section 2.1, Equation 2.1, Page 2-7. This equation is repeated below:

$$\forall i, 1 \leq i \leq n, min(\sum_{j=1}^{i-1} C_j \frac{1}{lT_k} \lceil \frac{lT_k}{T_j} \rceil + \frac{C_i}{lT_k} + \frac{B_i}{lT_k}) \leq 1$$

$$(k, l) \in R_i$$

where C_j and T_j are the execution time and period of task τ_j respectively, $R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \ldots, \lfloor \frac{T_i}{T_k} \rfloor\}$ and B_i is the worst case blocking time for τ_i.

The objective was to model blocking experienced from any source: the runtime system, other user tasks, I/O, etc.

In order to construct such an equation, detailed data had to be gathered on the blocking a task experiences while running on an actual processor. In order to gather that data without introducing timing errors into the measurements, as can happen when using software to gather such information, specialized hardware monitors are required. Unfortunately, that hardware was not available.

The only other approach available was to use a software based method to gather such information. The System Designers XD Ada runtime kernel included the ability to record the execution path while executing in the kernel, but there is a three-fold problem associated with that: (1) no time tags were attached to the execution trace, (2) even if there were, the overhead required to generate such tags might be sufficient to render the timing information useless, and (3) the data gathered from the kernel was output in real-time to the single board computer's serial port, thereby introducing an I/O latency that certainly rendered the data useless.

Consideration was given to overcoming these problems by adding the time tag information to the kernel execution trace and outputting the data to an area in memory for read out after execution. Of course, the problem of ensuring the time tag did not introduce excessive overhead would require that the approach be validated which, in turn, would require the specialized hardware which was not available in the first place.

Another difficulty was that the Hartstone benchmark provides CPU utilization data on the user task set only. The amount of unused CPU processing capacity that is system overhead and the amount that is CPU idle time is not provided. CPU idle time could be determined by creating a low priority user task which would run only when the CPU wasn't performing other tasks.

This task would record the amount of time spent within it. This amount of time would be the CPU idle time. The lack of this data meant that it would not be possible, using the Hartstone, to determine what portion of unused CPU capacity was system overhead and use that data to construct the mathematical equation for blocking. Data on system utilization of the CPU would provide additional insight into a processor's runtime behavior and would be a valuable addition to the Hartstone benchmark.

Given these difficulties, focus shifted to constructing a parameterized computational model of a runtime system. Timing data for the individual system tasks was gathered using the ACEC and existing documentation for the particular runtime system. The Hartstone benchmark served to validate the model, since, in contrast to the ACEC, it provides timing information on tasks as they execute under the runtime system. Additionally, since the computational model will control the system clock, an event trace and statistics can be provided without introducing any timing errors into the model. The event trace and timing statistics provide valuable insight into user task set interaction with a runtime system.

4.3 Model Requirements

The requirements for the RATESIM model provided the basis for the final design. The requirements were divided into three types: input requirements, functional requirements, and output requirements. These requirements are itemized below.

4.3.1 Input Requirements The RATESIM model must:

1) Have the ability to define a user task set large enough to represent those likely to be encountered in an actual system.

2) Accept the following task parameters:

   (a) worst case execution time,

(b) period,

(c) deadline, and

(d) synchronization points.

3) Have the ability to specify an unlimited number of system tasks along with their associated execution time and frequency.

4.3.2 Functional Requirements The model must:

1) Provide a 1 μs simulation time resolution.

2) Use a preemptive, event-driven, fixed priority scheduling algorithm,

3) Support the following tasking constructs: (a) task suspension, and (b) task synchronization.

4) Provide the following runtime system functions: (a) context switch, and (b) system clock update.

5) Assign priorities according to the rate monotonic priority assignment scheme.

6) Make a conservative determination of user task set schedulability (i.e. does not report a task set can successfully meet all its deadlines when in fact it cannot).

4.3.3 Output Requirements The model must:

1) Provide a time ordered history of runtime events,

2) Provide the following statistics on each user task:

(a) cumulative execution time,

(b) number of deadlines met,

(c) number of deadlines missed,

(d) time of first deadline missed,

(e) for the first deadline missed, the time execution associated with that deadline was finally completed,

(f) cumulative time of late deadlines,

(g) number of preemptions suffered due other tasks,

(h) cumulative time of early deadlines,

(i) number of context switches,

(j) and number of delay expirations.

3) Provide the following statistics for each simulation run:

(a) simulation time,

(b) user cumulative task execution time,

(c) user deadlines met,

(d) user deadlines missed,

(e) context switches,

(f) delay expirations,

(g) system task execution time,

(h) idle time,

(i) percentage user task execution time,

(j) percentage system task execution,

(k) percentage idle time,

(l) cumulative induced priority inversion time due to DELAY statement jitter.

4.3.4 RATESIM Specification Specifications for RATESIM are contained in Table 4.1.

The context diagram shown in Figure 4.1 illustrates how RATESIM interacts with the outside world. RATESIM requires the user to:

1) specify the user task set either by supplying the file where the task set is stored (Filename) or entering the task set interactively (Task Parameters),

2) specify the file where a task set is to be saved to (Filename),

3) and provide the length of time to run the simulation (Simulation Time).

4.3.5 Environmental Model This section describes the environment in which the RATESIM model exists. It specifies the purpose of RATESIM, the input data it requires, and the output data it provides.

The following list is a comprehensive list of user inputs which RATESIM will respond to.

1) add a task,

2) delete a task,

3) edit a task,

4) run simulation,

5) rate monotonic equation,

6) display task set,

7) get task set from a file,

8) save task set to a file,

9) and invalid input.

All of the above inputs are self-explanatory except rate monotonic equation. That causes RATESIM to determine whether or not the task set is schedulable based on Equation 2.1, on Page 2-7 (repeated below).

$$\forall i, 1 \leq i \leq n, min(\sum_{j=1}^{i-1} C_j \frac{1}{lT_k} \lceil \frac{lT_k}{T_j} \rceil + \frac{C_i}{lT_k} + \frac{B_i}{lT_k}) \leq 1$$
$$(k,l) \in R_i$$

where C_j and T_j are the execution time and period of task τ_j respectively, $R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \ldots, \lfloor \frac{T_i}{T_k} \rfloor\}$ and B_i is the worst case blocking time for τ_i.

Its purpose is to provide a confidence check of the simulation results. The simulation results and the calculated inequality should agree.

RATESIM provides to the user:

1) the result of the rate monotonic equation (Rate Monotonic Equation),

2) a chronological list of simulation events (Event History),

3) and the statistics gathered during the course of the simulation (Statistics).

4.4 RATESIM Design

This section describes the internal behavior of the RATESIM model. Eight entities are used within RATESIM and Figure 4.2 illustrates the relationships between them.

1. System Statistics is a data structure that holds all the system-wide statistical data collected during a given RATESIM simulation. It has no explicit relationship with any other entity.

2. Ready Queue is a prioritized queue (based on the task period) of User Tasks. Ready Queue is used during the execution of RATESIM to hold any User Task which is ready to execute but has not yet been granted access to the simulation "CPU".

3. System Queue is a prioritized queue (based on execution start time) of System Tasks. System Queue is used during the execution of RATESIM to hold any System Task which is ready to execute but whose execution time has not yet arrived.

4. System Task contains the parameters associated with a particular system task. Examples of the task parameters include task type and execution time. Note that a System Task may be implicitly defined within RATESIM such as a context switch or system clock update, or a System Task may be initiated by a User Task requesting a system service such as task suspension (or DELAY), or an Ada rendezvous (ACCEPT or ENTRY CALL). System tasks, in this version of RATESIM, are manually added by modifying the RATESIM source code.

5. Rendezvous Queue is a prioritized queue (based on arrival time) of ACCEPT or ENTRY System Tasks. Rendezvous Queue is used to hold an ACCEPT or ENTRY System Task executed on behalf of the User Task that made the call, but which has not yet received the corresponding ACCEPT or ENTRY call.

6. User Task contains all the parameters associated with a particular user-specified task. Examples of task parameters include period, execution time, and deadline. As seen in Figure 4.2 there is a one-to-one correspondence between an instance of User Task and an instance of User Statistics.

7. User Statistics is a data structure that holds all the statistical data collected during a given RATESIM simulation for a given User Task.

8. Rendezvous Ring is a prioritized ring (based on execution start time) of task synchronization events (an entry call or accept). After the execution of a call or accept the ring is rotated to point to the next synchronization event. After execution is complete for the given user task period, the ring is reset to point to the first synchronization event of the period

4.4.1 RATESIM Transaction Diagram and Data Flow Model The transaction diagram in Figure 4.3 shows the transactions that occur at the user interface level of RATESIM. Most of the RATESIM functions depend on the Task Parameters (supplied by the user) which are used to construct User Tasks and are then placed in the data store Task List. An external data store (e.g. a file) Tasks is used to store User Tasks between executions of RATESIM.

Figure 4.4 shows the data flow during the simulation. First, all User Tasks are placed in the store Ready Queue and the Simulation Run Flag is set to true. Placing all User Tasks on the Ready Queue establishes the worst case phasing of the user task set. Initial (or implicit) System Tasks such as a system clock update are read from the System Task store and placed in the store System Queue. System Tasks are also placed on System Queue during the simulation depending on what system services (e.g. task suspension, synchronization) are requested by a User Task. The Simulation Time is obtained from the user and simulation begins. During the course of the simulation user task statistics are updated and placed in the User Statistics store, the Event History is produced, system statistics are saved in the System Statistics store, and the System Queue and Ready Queue have user and system tasks added and removed as required.

For the Rate Monotonic Equation process, tasks are input from the User Task store (see Figure 4.5) and if the Simulation Run Flag is true, blocking information is read from the User Statistics. Schedulable is set to true or false based on the result of the Rate Monotonic Equation.

Figure 4.6 shows the Do Simulation process at a more detailed level during the simulation. Execute System Task takes a system task from the front of the System Queue and updates System Statistics, outputs an event (Event History) and may (depending on the system task) place a User Task on the Ready Queue. Similarly, Execute User Task takes a user task from the front of the Ready Queue and may (depending on the the user task) take a system task from the Rendezvous Queue. It then updates User Task Statistics/System Statistics, records an event (Event History) and may (again depending on the user task) place a System Task on the System Queue or

back on the Rendezvous Queue. Calculate Statistics occurs at the end of the simulation and it simply gathers the user task and system statistics generated during the simulation and outputs them to the user.

4.4.2 Flow Charts Much of the control flow in RATESIM occurs at the user interface level and is largely routine and uninteresting. Therefore, the following section will presents details about the Do System Task, Do User Task, and Do Idle processes within RATESIM. These processes are explained further in the sections below.

4.4.2.1 Do System Task RATESIM operates in the process Do System Task while there is a system task which has a start time less than or equal to the current simulation time. Two data structures define this state, System Task Queue and Time. System Task Queue is a prioritized queue of System Tasks with the start time of the task determining the priority. Time is simply an integer which contains the elapsed simulation time in microseconds.

When RATESIM begins, there is only one system task on System Task Queue, Clock Update. This system task is the only task that will execute independent of a user task requesting a system service. The other system tasks, Context Switch, Delay, Rendezvous Call, and Rendezvous Accept, are only initiated upon a user task request for the service and are explained below (see Figure 4.7).

1. Clock Update: When a Clock Update is executed, the following things occur:

   (a) the task is removed from the System Task Queue,

   (b) the clock update event is recorded,

   (c) since this is a periodic system task, the next Clock Update execution is added to the System Task Queue,

   (d) and finally Clock Update is "executed" by adding the Clock Update execution time to Time.

2. Context Switch: During a Context Switch, Do System Task performs the following actions:

   (a) the context switch event is recorded,

   (b) and Context Switch is "executed" by adding the Context Switch execution time to Time.

   A context switch can be modeled in many ways. One method of modeling it is to add twice the context switch execution time (once for switching into, then out of the user task) to the task execution time (8). Therefore, it becomes indistinguishable from the user task execution time. Another way to model it is as a system task that is executed each time the task begins execution. The distinction is that when the context switch is simply added to the user execution time, accounting for the time spent doing context switches is no longer possible. Therefore, RATESIM models a context switch as a system task that is executed each time the task begins execution. This design decision permitted cleaner design since it clearly separated what was user task utilization of the CPU and what was overhead (i.e. a system task).

   A by-product of this design decision is that the Context Switch task does not involve removing a task from the System Task Queue. That is, rather than a scheduled system task execution, a user task requests the context switch when it begins its execution and the system service is immediately performed.

   A Delay is the result of a User Task requesting suspension. The User Task that requested the suspension is saved by the Delay system task so that when the system task is executed, the User Task requesting the suspension can be rescheduled.

3. Delay: When a Delay is executed, the following things occur:

   (a) the user task requesting the suspension is rescheduled on the Ready Queue,

(b) the Delay task is removed from the System Task Queue,

(c) the delay event is recorded,

(d) Delay is "executed" by adding the Delay execution time to Time,

(e) if, after updating Time, any other Delays on System Queue have execution start times less than or equal to Time, they are also executed but at a significantly smaller execution time penalty.

This is an assumed optimization generally implemented in most runtime systems. That is, if some other task's delay will expire within the amount of time it takes to "wake up" a previous task's delay expiration, the other task will be rescheduled at a reduced penalty. In the case of the XD Ada runtime environment, the reduced penalty was 4 μs.

4. Rendezvous Call: A Rendezvous Call is executed when a User Task (the calling task) requests synchronization with another task (the accepting task). A single ASCII character is used to identify the Accept entry the calling task wants to synchronize with. Timed calls are not supported. That is, if a corresponding Accept (from the accepting task) is not executed, the calling task will be suspended forever. Although it is a system task, a Rendezvous Call will never be found on the System Task Queue. It is held on the Rendezvous Queue until the corresponding Accept is executed.

5. Rendezvous Accept: A Rendezvous Accept is executed when a User Task (the accepting task) has accepted synchronization from the calling task. Note that until both the Rendezvous Call and the corresponding Rendezvous Accept have been executed, the Call or Accept will reside in the Rendezvous Queue. After both have been executed, a Rendezvous Accept system task will be placed on the System Task Queue. Upon executing a Rendezvous Accept the following will occur:

(a) the accepting task's priority will be changed to the higher of its own priority or the calling task priority,

(b) the accepting task will be rescheduled,

(c) the Rendezvous Accept task is removed from the System Task Queue,

(d) the rendezvous event is recorded,

(e) and Rendezvous Accept is "executed" by adding the Rendezvous Accept execution time to Time.

RATESIM enters Do User Task when the next system task to execute has a start time greater than Time. Two data structures define Do User Task , Ready Queue and Time. Time is the same data structure as described in the previous section. Ready Queue is a prioritized queue of User Tasks with priorities assigned according to RMA.

When Do User Task begins, there are zero or more User Task's on the queue. There may be zero tasks on the queue for one of two reasons: (1) no user tasks were defined, or (2) all user tasks that were defined have requested a system service and are either on System Task Queue or Rendezvous Queue. If there are zero User Task's on the queue RATESIM enters the Do Idle state (described below) until such time as the next System Task is executed.

In Do User Task, the following sequence of events occur (see Figures 4.8 and 4.9):

1) CHECK NEXT USER TASK (1): if Next System Task Start = TIME or Ready Queue is empty then no User Task is executed.

2) DO CONTEXT SWITCH (2): a Context Switch system task is executed unless this User Task is being executed twice in a row without a different User Task or System Task being executed in between.

   This situation could occur immediately after task synchronization when both tasks that were in rendezvous are rescheduled. The accepting task (which was executing)

might still be the highest priority User Task and begin execution again. In this situation, no Context Switch system task is initiated.

3) BEGIN EXECUTION (3): the time available for the User Task execution is determined (the next system task start time minus the current time).

4) the User Task is removed from the Ready Queue.

5) a User Task Execution Begin event is recorded.

6) ADJUST AVAILABLE TIME (4): if the User Task is in the middle of an Accept execution, the time available for User Task execution is adjusted to be the smaller of the current available time (as determined in item ii above) or the Accept execution time.

7) EXECUTE TASK (5): the User Task is "executed".

User Task execution contains many substates which are described in Section 4.4.2.2.

8) EXECUTION END (6): a User Task Execution Stop event is recorded.

9) REQUEST DELAY (7): if the User Task has completed its execution time for this period, a Deadline Met or a Deadline missed event is recorded and the User Task initiates a task suspension system task (Delay).

10) RESCHEDULE (9): if the User Task has not completed its execution time for this period, it is rescheduled.

11) SEARCH FOR ENTRY OR ACCEPT (8): if the User Task has executed a Rendezvous Entry or Rendezvous Accept the Rendezvous Queue is searched for a corresponding entry or accept, and a Rendezvous Event is recorded.

CREATE RENDEZVOUS SYSTEM TASK (10): if the corresponding entry is found, the User Task initiates a Rendezvous Accept system task.

PLACE ON RENDEZVOUS QUEUE (11): if not found, the User Task is placed on the Rendezvous Queue to await the corresponding entry or accept.

12) UPDATE NEXT SYSTEM TASK START (12): finally, since User Task's can initiate a system task which may have a start time earlier than the current next system task start time, the next system task start time is updated.

4.4.2.2 Execute User Task The Execute User Task state (see Figure 4.10) is part of the Do User Task state. Due to the many substates within Execute Task, it is described in detail in this section.

When Execute User Task begins, the following information is passed to it: (1) Available Time (for task execution) and (2) The Task (the task to execute). The following sequence of events then occurs (see Figure 4.10):

1) CHECK AVAILABLE TIME (5.1): it is determined whether a rendezvous (entry call or accept) will occur during this execution.

2) Available Time is checked to determine: (a) if it is zero or less, (b) whether The Task can complete its execution within Available Time.

   Available Time could be zero or less if the Context Switch execution time used up all the execution time budget. If so, no execution time is allotted to The Task.

   If The Task will complete its execution within Available Time, an execution complete flag is set.

3) ALLOCATE EXECUTION TIME (5.2): if a rendezvous will occur during this execution time, sufficient execution time is allocated to The Task to reach the rendezvous point, otherwise up to Available Time execution time is allocated to The Task.

4) ALLOCATE ACCEPT TIME (5.3): if The Task is in the process of executing an Accept (it was already in rendezvous when execution began), then execution time is allocated to the Accept execution time.

5) RECORD STOP ACCEPT (5.4): if the Accept has been allocated sufficient execution time to finish, a Stop Accept event is recorded and the task that placed the entry call is rescheduled.

6) RECORD EXECUTION STOP (5.5): a User Task Execution Stop event is recorded.

7) RECORD EXECUTION TIME REQUIRED (5.6): and finally an Additional User Task Execution Time Required event is recorded.

4.4.2.3 Do Idle When Time is less than the next system task start time and there are no User Tasks to execute, RATESIM transitions to Do Idle and an Idle event is recorded. Do Idle is terminated when the next system task start time is equal to Time.

4.5 Testing

This section addresses the testing RATESIM underwent to ensure proper operation. Contrasted with validation (discussed in the next chapter) which attempts to ensure that the RATESIM ***model design*** accurately simulates an embedded runtime system, testing ensures the proper operation of the RATESIM ***program*** .

The objective in testing RATESIM was to give a reasonable assurance that the program would not abort execution abnormally due to invalid user input or lack of operating system memory. Additionally, to the extent that data integrity checks were performed by RATESIM, the objective was to ensure that those checks operate correctly.

The data integrity checks performed by RATESIM to ensure user entered data is of the proper form does not include all checks necessary to ensure the integrity of the user task parameters input

to RATESIM. For instance, when inputing rendezvous events, RATESIM depends on the user to input them sequentially from the earliest start time to the latest. If the user does not, the result will be unpredictable. The additional data integrity checks that need to be added to the RATESIM program are listed in Table 4.3.

4.5.1 Test Cases This section enumerates the test cases that were used to test RATESIM and the results of those tests. Data integrity tests that ensure proper operation of RATESIM but currently rely on proper user input are listed, but the test results are marked N/I (not implemented).

Three classes of test were performed: (1) user input, (2) data integrity, and (3) stress tests. User input tests will ensure that no user input will cause the program to abort abnormally. Data integrity tests will ensure that the data input is in a form that RATESIM is expecting. Stress tests will ensure that RATESIM will perform as designed at the limits of its specifications.

4.5.2 Event History Example The following is an sample of the output of RATESIM's event history. System Tasks events are in uppercase for easier identification. Table 4.5 lists all the events recorded by RATESIM and the information reported when those events occur such as start time, stop time, and execution time.

```
User Task Task 4(PID  2) started executing at    6385952.
User Task Task 4(PID  2) stopped executing at    6388944.
   Execution time :    2992 us.
User Task Task 4(PID  2) still requires              0 us of execution time.
User Task Task 4(PID  2) met its deadline of     6400136 at 6388944.
   It was  11192 us early.
User Task Task 4(PID  2) requested a DELAY of :     11192 us.
   Actual DELAY will be :     11213 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6388944 TO   6389093.
   EXECUTION TIME :     149 US.

User Task Task 2(PID  4) started executing at    6389093.
User Task Task 2(PID  4) stopped executing at    6392141.
   Execution time :    3048 us.
User Task Task 2(PID  4) still requires            5817 us of execution time.
SYSTEM_TASK A_DELAY        FROM   6392141 TO   6392300.
```

```
   EXECUTION TIME :     159 US.
DELAY EXPIRATION OF Task 5(PID  1).
SYSTEM_TASK CONTEXT_SWITCH FROM   6392300 TO   6392449.
   EXECUTION TIME :     149 US.

User Task Task 5(PID  1) started executing at    6392449.
User Task Task 5(PID  1) stopped executing at    6393945.
   Execution time :    1496 us.
User Task Task 5(PID  1) still requires              0 us of execution time.
User Task Task 5(PID  1) met its deadline of     6400136 at 6393945.
   It was   6191 us early.
User Task Task 5(PID  1) requested a DELAY of :      6191 us.
   Actual DELAY will be :      6175 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6393945 TO   6394094.
   EXECUTION TIME :     149 US.

User Task Task 2(PID  4) started executing at    6394094.
User Task Task 2(PID  4) stopped executing at    6399911.
   Execution time :    5817 us.
User Task Task 2(PID  4) still requires              0 us of execution time.
User Task Task 2(PID  4) met its deadline of     6432624 at 6399911.
   It was  32713 us early.
User Task Task 2(PID  4) requested a DELAY of :     32713 us.
   Actual DELAY will be :     32825 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6399911 TO   6400060.
   EXECUTION TIME :     149 US.

User Task Task 1(PID  5) started executing at    6400060.
User Task Task 1(PID  5) stopped executing at    6400120.
   Execution time :      60 us.
User Task Task 1(PID  5) still requires           23878 us of execution time.
SYSTEM_TASK A_DELAY        FROM   6400120 TO   6400279.
   EXECUTION TIME :     159 US.
DELAY EXPIRATION OF Task 5(PID  1).
SYSTEM_TASK A_DELAY        FROM   6400279 TO   6400283.
   EXECUTION TIME :       4 US.
DELAY EXPIRATION OF Task 4(PID  2).
SYSTEM_TASK A_DELAY        FROM   6400283 TO   6400287.
   EXECUTION TIME :       4 US.
DELAY EXPIRATION OF Task 3(PID  3).
SYSTEM_TASK CONTEXT_SWITCH FROM   6400287 TO   6400436.
   EXECUTION TIME :     149 US.

User Task Task 5(PID  1) started executing at    6400436.
User Task Task 5(PID  1) stopped executing at    6401932.
   Execution time :    1496 us.
User Task Task 5(PID  1) still requires              0 us of execution time.
User Task Task 5(PID  1) met its deadline of     6408258 at 6401932.
   It was   6326 us early.
```

```
User Task Task 5(PID  1) requested a DELAY of :      6326 us.
   Actual DELAY will be :      6500 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6401932 TO   6402081.
   EXECUTION TIME :     149 US.

User Task Task 4(PID  2) started executing at    6402081.
User Task Task 4(PID  2) stopped executing at    6405073.
   Execution time :    2992 us.
User Task Task 4(PID  2) still requires              0 us of execution time.
User Task Task 4(PID  2) met its deadline of     6416380 at 6405073.
   It was  11307 us early.
User Task Task 4(PID  2) requested a DELAY of :     11307 us.
   Actual DELAY will be :     11375 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6405073 TO   6405222.
   EXECUTION TIME :     149 US.

User Task Task 3(PID  3) started executing at    6405222.
User Task Task 3(PID  3) stopped executing at    6406400.
   Execution time :    1178 us.
User Task Task 3(PID  3) still requires           4806 us of execution time.
SYSTEM_TASK CLOCK_UPDATE   FROM   6406400 TO   6406415.
   EXECUTION TIME :      15 US.
SYSTEM_TASK CONTEXT_SWITCH FROM   6406415 TO   6406564.
   EXECUTION TIME :     149 US.

User Task Task 3(PID  3) started executing at    6406564.
User Task Task 3(PID  3) stopped executing at    6408432.
   Execution time :    1868 us.
User Task Task 3(PID  3) still requires           2938 us of execution time.
SYSTEM_TASK A_DELAY        FROM   6408432 TO   6408591.
   EXECUTION TIME :     159 US.
DELAY EXPIRATION OF Task 5(PID  1).
SYSTEM_TASK CONTEXT_SWITCH FROM   6408591 TO   6408740.
   EXECUTION TIME :     149 US.

User Task Task 5(PID  1) started executing at    6408740.
User Task Task 5(PID  1) stopped executing at    6410236.
   Execution time :    1496 us.
User Task Task 5(PID  1) still requires              0 us of execution time.
User Task Task 5(PID  1) met its deadline of     6416380 at 6410236.
   It was   6144 us early.
User Task Task 5(PID  1) requested a DELAY of :      6144 us.
   Actual DELAY will be :      6175 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6410236 TO   6410385.
   EXECUTION TIME :     149 US.

User Task Task 3(PID  3) started executing at    6410385.
User Task Task 3(PID  3) stopped executing at    6413323.
```

```
   Execution time :     2938 us.
User Task Task 3(PID  3) still requires                0 us of execution time.
User Task Task 3(PID  3) met its deadline of      6432624 at 6413323.
   It was  19301 us early.
User Task Task 3(PID  3) requested a DELAY of :      19301 us.
   Actual DELAY will be :      19338 us.

SYSTEM_TASK CONTEXT_SWITCH FROM   6413323 TO   6413472.
   EXECUTION TIME :      149 US.

User Task Task 1(PID  5) started executing at    6413472.
User Task Task 1(PID  5) stopped executing at    6416411.
   Execution time :     2939 us.
User Task Task 1(PID  5) still requires            20939 us of execution time.
```

4.6 Summary

The need for specialized measurement hardware or expensive simulator development to characterize a single runtime environment motivated the development of a software-based, parameterized model that could simulate various runtime environments. The purpose of the RATESIM model is to simulate an embedded runtime system that will execute a given task set and monitor the user task set interaction with the runtime system. The RATESIM model has two objectives: (1) to determine whether a user task set can execute on a real processor with its associated runtime system without missing any of its deadlines, and (2) to provide insight into the user task set interaction with that runtime system. This chapter also presented the requirements and design of the RATESIM program. In addition, the testing RATESIM underwent was discussed.

Table 4.1. RATESIM Specification

Item	Range
User Tasks	0 to 99
System Tasks	no limit
Simulation Time (in μs)	0 to 2,100,000,000 (0 to 35 minutes)
Synchronization Events (Entry Call or Accepts)	0 to 20 per user task
Synchronization Point Names	single alpha-numeric character

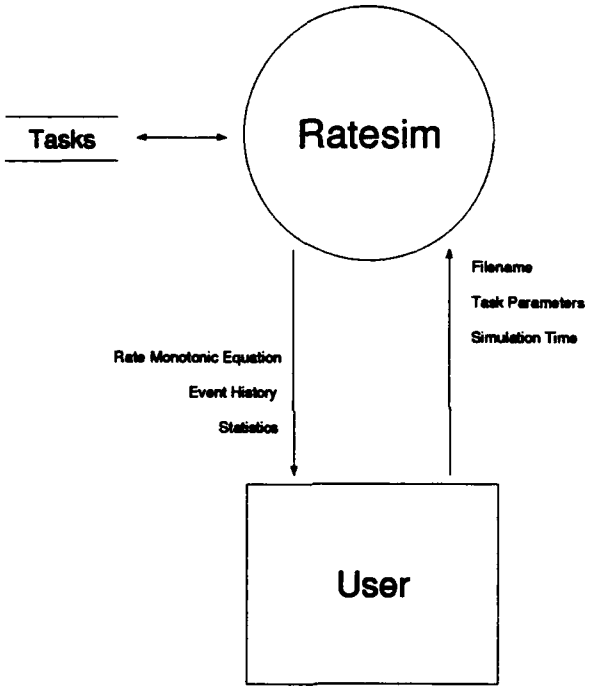

Figure 4.1. RATESIM Context Diagram

Table 4.2. Test Cases - User Input

Test	Objective	Result
Main Menu/Valid Input	accept valid menu choice	pass
Main Menu/Invalid Input	reject invalid menu choice	pass
Numeric Input/Valid Input	accept valid numeric input	pass
Numeric Input/Invalid Input	reject invalid or out of range numeric input	pass
Text Input/Valid Input	accept valid text	pass
Text Input/Invalid Input	reject strings which are too long	pass

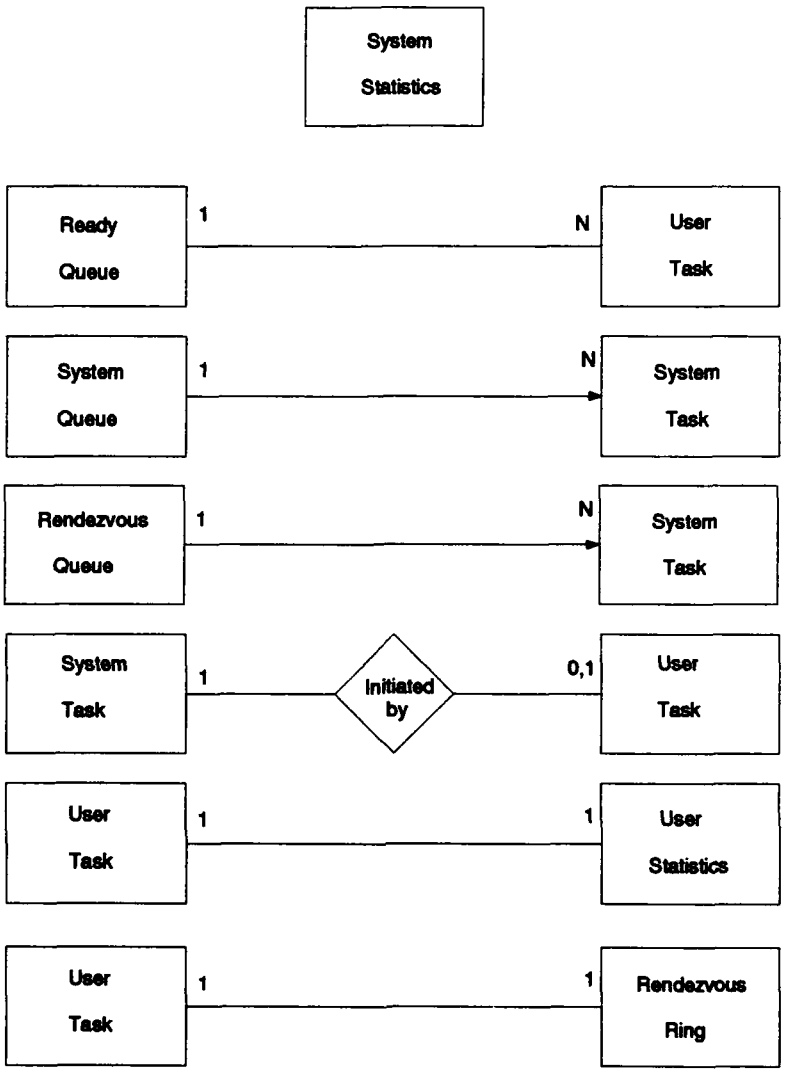

Figure 4.2. RATESIM Entity Relationships

Table 4.3. Test Cases - Data Integrity

Test	Objective	Result
User Task Period	ensure period $\geq$ execution time	pass
User Task Deadline	ensure deadline $\geq$ execution time and $\leq$ period	pass
Rendezvous Order	rendezvous points sequential from earliest to latest	N/I
Rendezvous Start	$0 \leq$ start time $\leq$ execution time	N/I
Rendezvous Points	rendezvous events do not overlap	N/I
Accept Execution Time - 1	ensure accept execution time + accept start time $\leq$ execution time	N/I
Accept Execution Time - 2	accept execution time ≥ 0	N/I
Simulation Time ≤ 0	reject invalid simulation time	pass

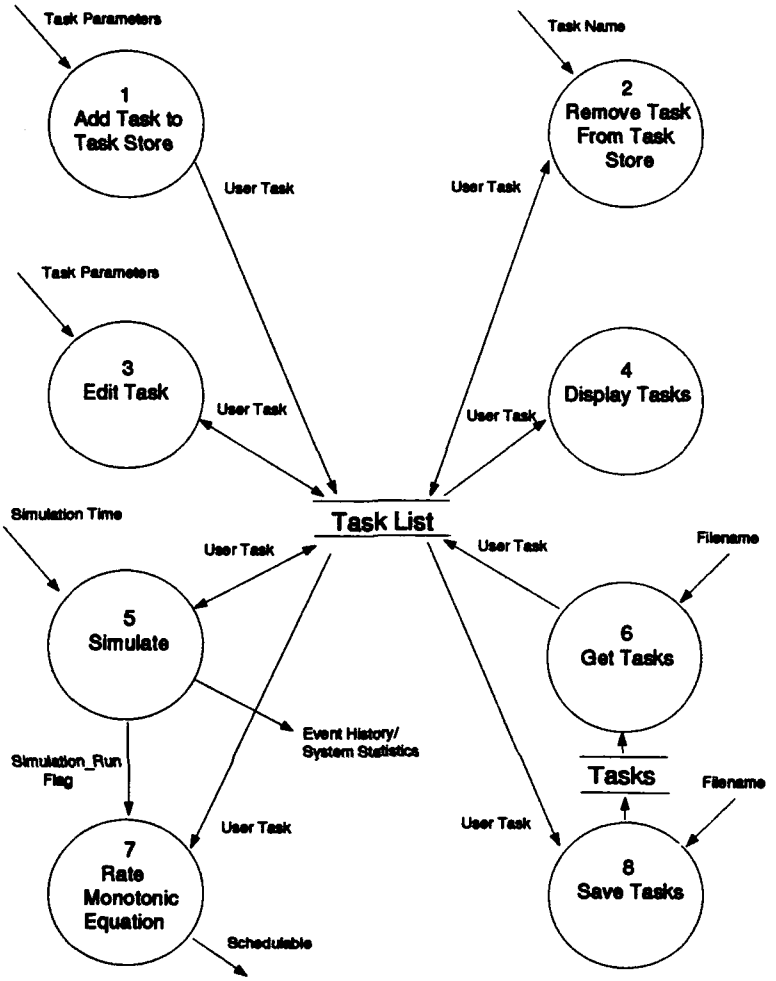

Figure 4.3. Transaction Diagram

Table 4.4. Test Cases - Stress Tests

Test	Objective	Result
User Tasks = 99	accept maximum # of user tasks	pass
User Tasks = 0	accept minimum # of user tasks	pass
Simulation Time - 1	proper operation at maximum simulation time	pass
Simulation Time - 2	proper operation at minimum simulation time	pass

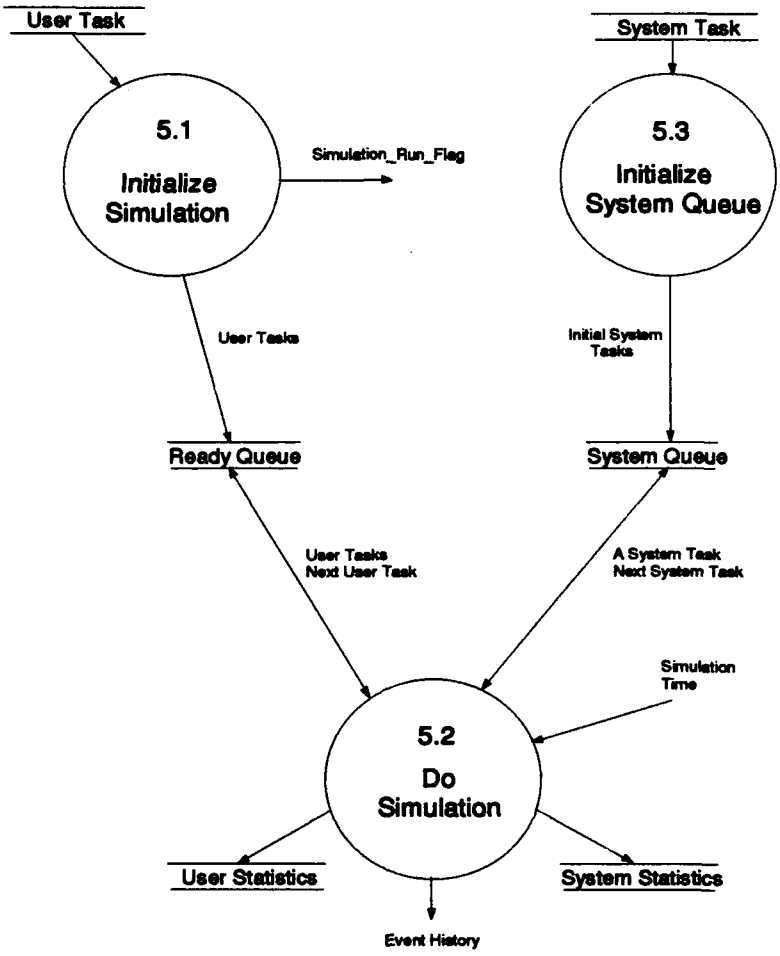

Figure 4.4. Level 2 DFD

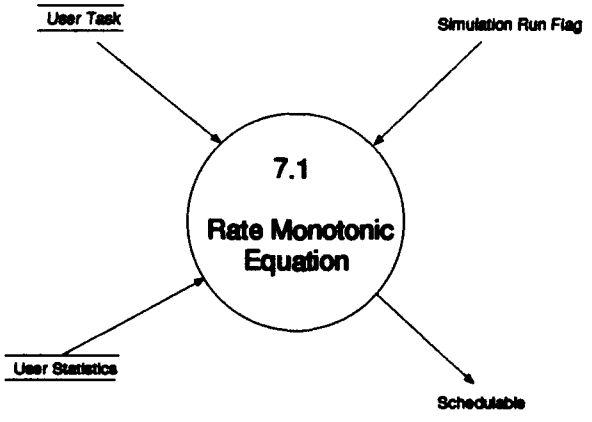

Figure 4.5. Level 2 DFD

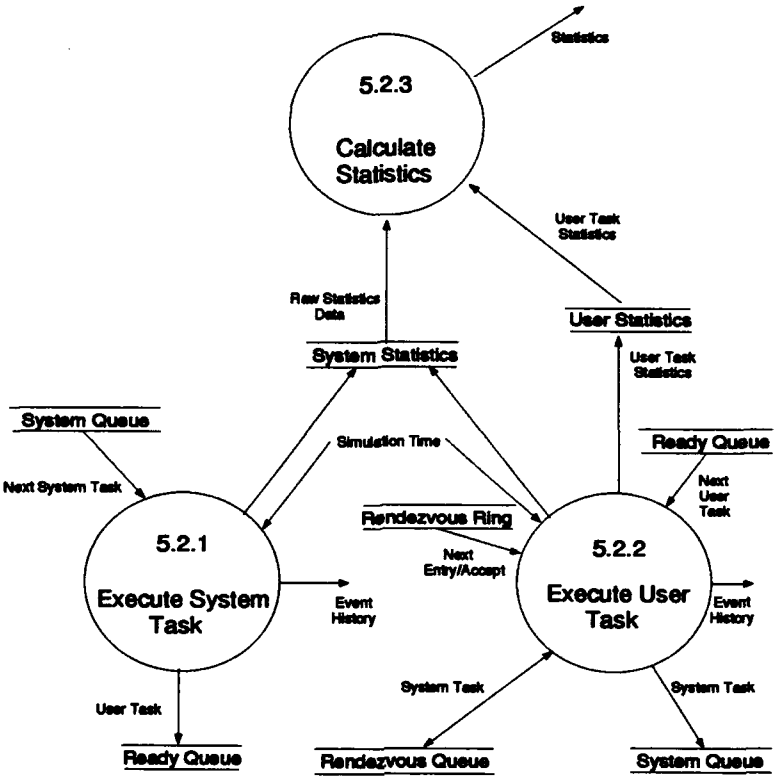

Figure 4.6. Level 3 DFD

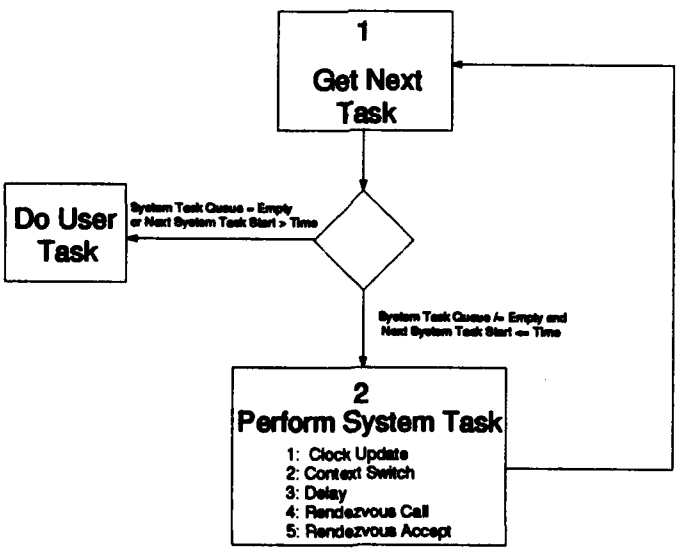

Figure 4.7. Do System Task

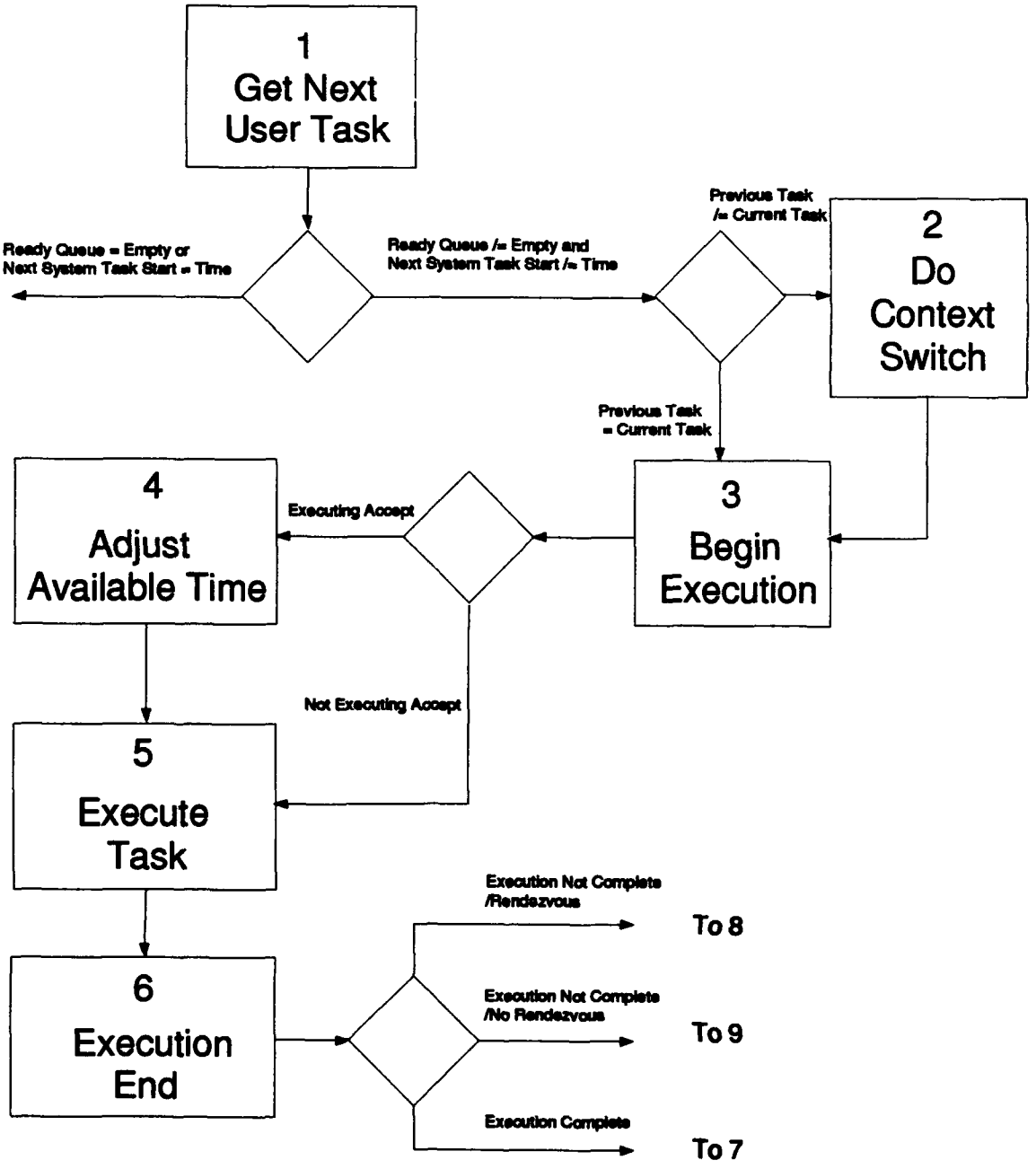

Figure 4.8. Do User Task

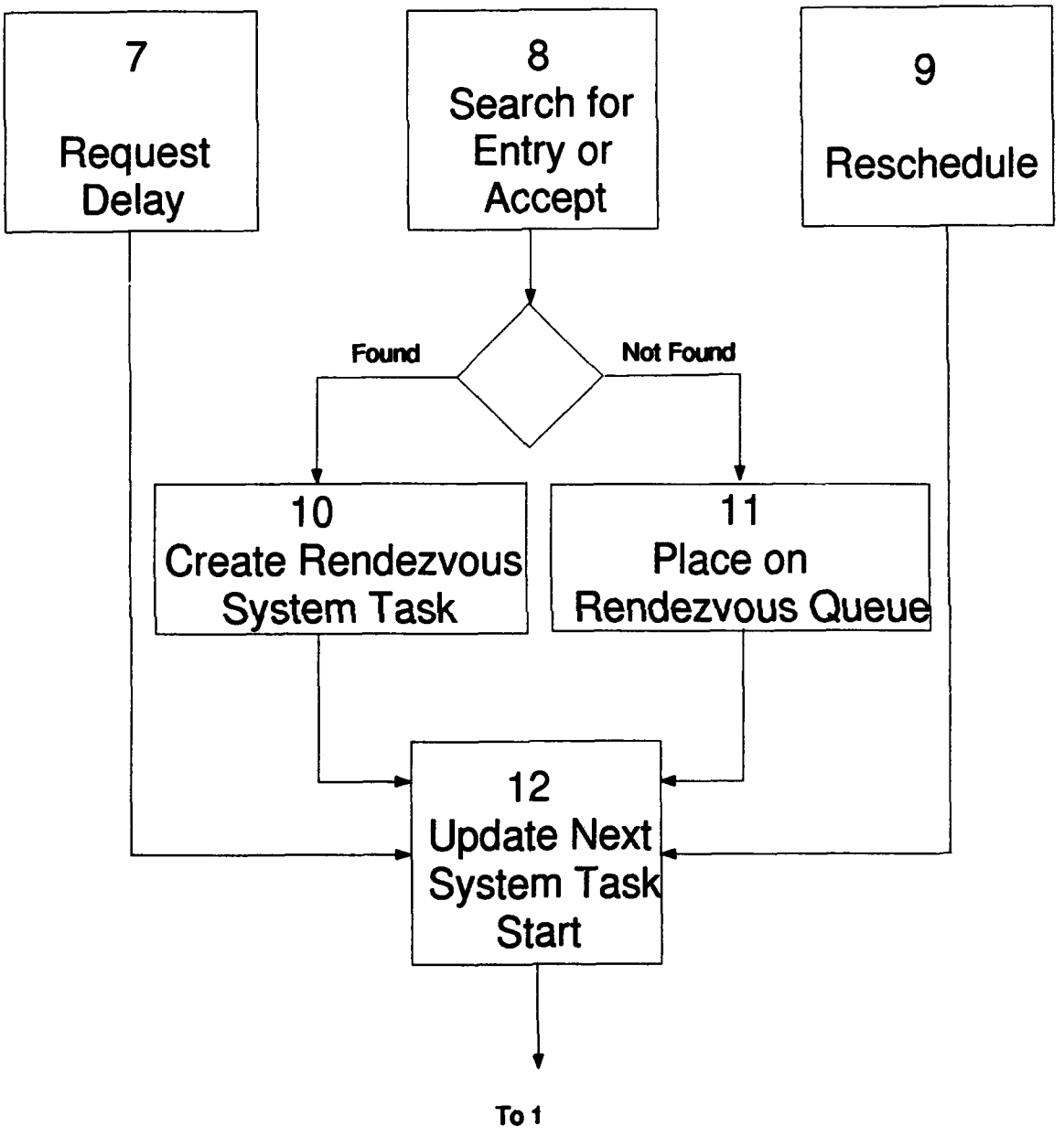

Figure 4.9. Do User Task(cont)

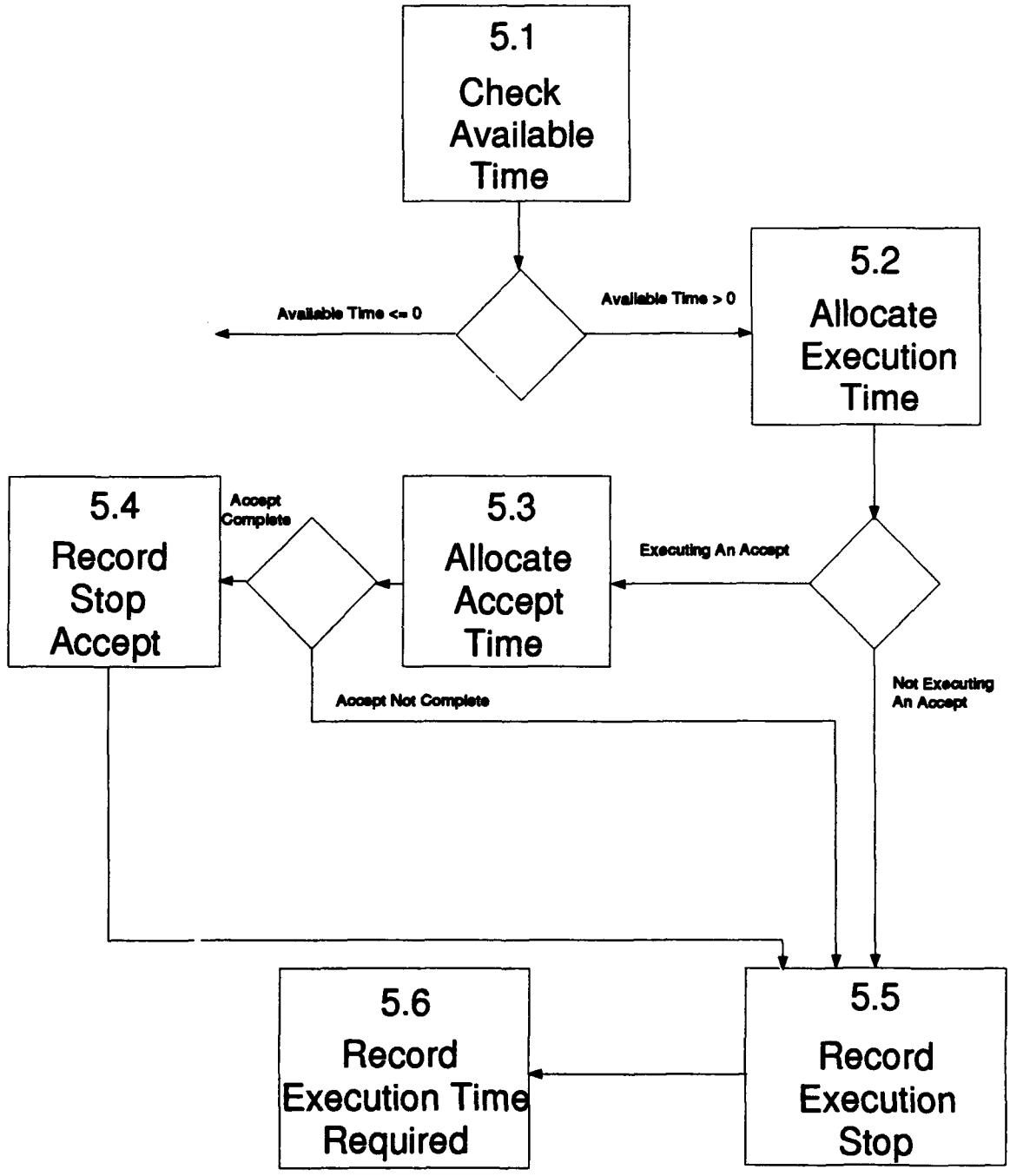

Figure 4.10. Execute Task

Table 4.5. RATESIM Events

Event	Event Information Supplied
Simulation_Begin	NONE
Simulation_End	NONE
User_Task_Execution_Start	User Task Name, User Task Process Identifier (PID), Execution Start Time
User_Task_Execution_Stop	User Task Name, User Task Process Identifier (PID), Execution Stop Time, Execution Time
User_Task_Execution	User Task Name, User Task Process Identifier (PID), Execution Time Still Required
User_Deadline_Met	User Task Name, User Task Process Identifier (PID), Current Deadline, Execution Complete Time, Microseconds Early
User_Deadline_Missed	User Task Name, User Task Process Identifier (PID), Current Deadline, Execution Complete Time, Microseconds Late
System_Task_Execution	System Task Name, Start Time, Stop Time, Execution Time, *(User Task Name, User Task Process Identifier (PID))
User_Task_Rescheduling	NONE
Idle	Start Time, Stop Time, Idle Time
Delay_Request	User Task Name, User Task Process Identifier (PID), Delay Request, Actual Delay
Call_Request	User Task Name, User Task Process Identifier (PID), Time of Call
Accept_Request	User Task Name, User Task Process Identifier (PID), Time of Accept
Stop_Accept	Called Task Name, Called Task Process Identifier (PID), Calling Task Name, Calling Task Process Identifier (PID), Rendezvous Complete Time
BAD_TIME	Current Time

* If a Delay request – User Task requesting the Delay

V. *RATESIM Validation*

The test cases that the RATESIM model was validated against were described in Chapter III. To summarize: the ACEC program was used to collect data on the target runtime environment such as context switch time, DELAY expiration jitter, and other pertinent system tasks. The ACEC data supplied the parameters for the system tasks modeled by RATESIM. Finally, various user task sets were constructed (based on the SEI Hartstone benchmark) and run on both the RATESIM model and the target hardware. The results were compared to further refine the accuracy of the RATESIM failure prediction.

The accuracy of the results obtained from the ACEC, the SEI Hartstone Benchmark, and manual code analysis of portions of the runtime source code directly influenced the validation of RATESIM. If these results are not valid, then the RATESIM model cannot hope to be valid. However, establishing the accuracy of these various measurement tools is not within the scope of this research and so the validation of RATESIM is based on the assumption that the measurement tools used to validate it are correct.

35Before presenting the Hartstone test cases and results used to validate RATESIM as a whole, the validation of some individual components of RATESIM is discussed. They are: the DELAY model (task suspension), system clock update, the scheduling algorithm, and context switch/rendezvous (task synchronization).

5.1 Delay Model

The ACEC test suite measures the runtime environment to determine the additional amount of time (T_a) a user task is suspended (over and above the requested delay) versus the suspension time the user task requested (T_r). The tests, as written, request DELAYs (T_r) of $0\ \mu s, 1\ \mu s, 10\ \mu s, 100\ \mu s, \ldots, 100000\mu s$. These tests determined that T_a ranged from 208.40 to $446.80\mu s$ for a given T_r. This effect, DELAY statement jitter, is a significant source of blocking to

user tasks. A higher priority task can, in effect, be prevented from executing when it is otherwise eligible to.

Note that this particular source of blocking, on the target hardware used during this research, is due to the resolution of the underlying hardware timers and not due to system resource constraints or a particular scheduling strategy. This being the case, it is not possible to generalize the behavior of a runtime system DELAY unless the target hardware uses the same design. Therefore, RATESIM embedded the DELAY jitter algorithm in an Ada function call. This facilitates the support of various DELAY implementations by encapsulating the DELAY jitter effect in one area of the RATESIM model.

Initially, RATESIM simply added the worst case T_a to T_r to calculate the worst case length of a user task DELAY request. This approach resulted in an extremely conservative prediction of task set failure. RATESIM would predict a task set failure well before the actual failure (as observed on the target hardware). Although a conservative prediction of task set failure was a design criteria for RATESIM, the results obtained were too conservative. It was necessary, therefore, to identify the sources of the additional delay and attempt to model their behavior so as to ultimately achieve a more accurate prediction. The details of the development of the DELAY model that was finally used is found in Appendix A and is summarized below.

In order to construct a DELAY model, more data than was provided by the ACEC was needed. Therefore, the ACEC delay test was modified to request DELAYs starting at $0\mu s$, and increase the DELAY request value by a specified amount such as $1\mu s, 10\mu s$, etc. The data collected is presented in tabular form in Appendix A, Section A.6.

The DELAY implementation of the runtime system (System Designer's XD Ada) converted the user task DELAY request twice before the hardware timers were set with a DELAY value. First, the DELAY request was converted from type REAL to type DURATION. Next it is converted from type DURATION to SYSTEM.TICK. The error introduced due to these conversions is accounted

for in the DELAY model used in RATESIM. By taking the conversion error into account, the DELAY statement jitter was reduced from a maximum of $446.80\mu s$ to a maximum of $304.40\mu s$. Of the remaining $304.40\mu s, 149\mu s$ can be attributed to the task context switching and is assumed constant. Therefore, the maximum DELAY statement jitter is further reduced to $155.40\mu s$. This remaining jitter is added to each DELAY request. The reduction of DELAY statement jitter means that up to $291.4\mu s$ of unnecessary blocking in the RATESIM model (depending on the DELAY request) is avoided and results in a more accurate prediction while still preserving the conservative nature of that prediction.

5.2 System Clock Update

The updating of the system clock is a fundamental function provided by any embedded runtime environment that is used in the real-time domain. Although the execution time of this function is typically very short, it nevertheless consumes CPU time, and given the correct task phasing, could cause a user task to miss a deadline. Therefore, it was important to account for this system task in RATESIM.

The ACEC does not provide a test to determine the amount of CPU time a clock update consumes. However, the amount of source code associated with the clock update function was small enough to perform a manual analysis of the code and determine the worst case execution time based on the published worst case instruction execution times of the MC68020 and the worst case execution path in the source code. This analysis assumed that the clock update function was never suspended due to higher priority interrupts. That is, once the clock update began, it executed to completion.

Appendix B contains the analysis of the clock update function. The execution time of the XD Ada clock update was determined to be $15.4\mu s$ every $41,600\mu s$.

5.3 Scheduling Algorithm

The scheduling algorithm used in RATESIM models a preemptive, event-based, fixed priority scheduler. Preemptive means that a higher priority task that is ready to execute will cause the suspension (or interruption) of the execution of any user task with a lower priority. Event-based means that the scheduling decisions are made at the time a given event occurs (i.e. a DELAY expiration or the completion of an interrupt service routine). Fixed priority means that the priority of user tasks do not change during their execution. One exception to the fixed priority rule occurs when two user tasks rendezvous (or synchronize). In this case, the tasks will execute at the higher of the two task's priorities.

In contrast to the other sections within this chapter which contain specific data and analysis on the particular aspect of the behavior in question to demonstrate its validity, this section takes a different approach. It will rely on the descriptions provided below coupled with the results of the Hartstone benchmark (see Section 5.5) as validation of correct behavior.

To the reader who would like to inspect the code within RATESIM which implements the scheduling behavior described above, the task priorities assignment code is contained in the procedure Utility_Body, procedures ADD TASK, and EDIT TASK. Scheduling decision code is embedded throughout the code contained in the procedure Simulate_Body.

5.3.1 Task Priorities User tasks priorities are assigned based on RMA. Thus, higher rate tasks are assigned a higher priority. The number of priority levels is equal to the number of tasks in the user task set. That is, no two tasks that would otherwise have different priorities will be forced to share the same priority due to a fixed number of priority levels.

Tasks having the same priority (due to having the same rate) are scheduled on a FCFS basis. Further, a task that is preempted before it completes its execution for a given period will be placed *ahead* of user tasks with the same priority. This prevents tasks of the same priority being served

on a round-robin basis and preserves the first-come-first-served (FCFS) nature of the scheduling algorithm for tasks of the same priority.

System tasks are modeled as a source of blocking to user tasks. They have no assigned priority, and will execute to completion once started.

5.3.2 Scheduling Decisions Scheduling decisions, for user tasks, are made after the completion of every system task execution (except the context switch). If there are more system tasks ready to execute, scheduling decisions for the ready user tasks are held off until such time that no more system tasks are ready. System tasks that result in a scheduling decision after execution include:

1) clock update,

2) DELAY expiration,

3) Rendezvous Call,

4) and a Rendezvous Accept.

System tasks are executed on a FCFS basis. Even though they have no assigned priorities to distinguish importance or urgency among other system tasks, once they are ready to execute, they preempt the current user task and then be modeled as a low priority user task which is blocking all other user tasks in the task set.

5.4 Context Switch and Rendezvous

The context switch and the rendezvous execution times are measured directly by the ACEC. Validation of RATESIM behavior for these two system tasks is limited to a description of the method in which the data was collected (19:A126-A130). ACEC output for the context switch and rendezvous tests can be found in ACEC pretest report (Appendix E).

5.4.1 Context Switch The context switch is measured within the ACEC using the following method (ACEC test problems – tk_lf_task_60, tk_lf_task_61, tk_lf_task_62):

a) measure the time required to increment a counter – *INCREMENT_COUNTER*,

b) measure the time required to execute a FOR loop *NUMBER_OF_CALLS* times (FOR loop body contains a DELAY 0.0) – *ZERO_DELAY*,

c) reset the counter,

d) measure the time required to execute a FOR loop *NUMBER_OF_CALLS* times (FOR loop body contains a DELAY 0.01) – *NONZERO_DELAY*,

e) execute the FOR loop in item ii above *NUMBER_OF_CALLS* times in a high-priority task, while a lower priority task increments the counter,

f) save the counter value in *SAVE_INCREMENT_COUNT*,

g) and finally, the estimated context switch time is:

$$\frac{NONZERO\_DELAY - ZERO\_DELAY - INCREMENT\_COUNTER \times SAVE\_INCREMENT\_COUNT}{NUMBER\_OF\_CALLS \times 2}$$

The variable *NONZERO_DELAY* contains the following system overhead components: context switch time and the delay overhead. The variable *ZERO_DELAY* contains the system overhead component for the delay. The variable *INCREMENT_COUNTER* and *SAVE_INCREMENT_COUNT* contains the CPU time used when not executing the high priority task *NONZERO_DELAY*. Finally, *NUMBER_OF_CALLS* contains the number of times *NONZERO_DELAY* was called and therefore, *INCREMENT_COUNTER*. After subtracting out the delay overhead and the time spent in the lower priority task, then dividing by the *NUMBER_OF_CALLS* x 2 (the number of context switches into *NONZERO_DELAY* and *INCREMENT_COUNTER*) the result will be an estimate of the context switching time.

5.4.2 Rendezvous Two measurements are made for simple Ada rendezvous in the ACEC: (1) the task making the entry call arrives first, and (2) the task doing the accept arrives first (ACEC test problems tk_lf_task_03, tk_lf_task_23). The ACEC does this by assigning a higher priority to the task which is supposed to arrive first.

The following is the code fragment of the test tk_lf_task_03 (tk_lf_task_23 is similar):

```
TASK resource IS
    PRAGMA priority( zg_glob1.priority_1 );
  ENTRY request;
  ENTRY release;
END resource;

TASK BODY resource IS
BEGIN
  LOOP
    ACCEPT request;
    ACCEPT release;
  END LOOP;
END resource;

TASK BODY main IS
BEGIN

  FOR i IN 1 .. 10 LOOP
    resource.request;
    resource.release;
  END LOOP;
```

It is possible, in a given runtime environment, that the execution time of a rendezvous will differ depending on whether the calling or the accepting task arrives first. RATESIM uses the greater of the two execution times.

5.5 Test Cases

The criteria for success when running the validation test cases consisted of two factors: (1) that RATESIM would predict the failure of the user task set prior to actual failure observed when

executing on the target hardware, and (2) that the failure mode (or manner in which the task set failed) would be similar. For example, if the user task set's failure on the target hardware was manifest by Task 5 missing ten deadlines, RATESIM should also predict that the failure would be from Task 5 missing deadlines.

The failure modes cited in the tables below are the failure modes from the *first* failure of the schedule according to RATESIM. Included in the raw data in Appendix C is data from RATESIM using the same task parameters that caused a failure using the Hartstone. Also note that Hartstone includes the number of deadlines skipped. Skipped deadlines occur in Hartstone when a task misses a deadline and attempts to "catch up" by load shedding (skipping deadlines) until the task determines it is possible to meet the next deadline. RATESIM does not incorporate load shedding since that is the responsibility of the user task. A runtime environment typically has no knowledge of whether or not an application task has missed a deadline. Therefore, no skipped deadlines will appear in the RATESIM failure modes.

A total of eight validation tests were run on RATESIM. The first three contained harmonic task sets, the next three contained non-harmonic task sets, and the final two had a harmonic task set that included synchronization. A summary of the results is contained in the following sections. The actual output of the Hartstone and RATESIM programs is contained in Appendix C.

The Hartstone benchmark is designed such that the parameter varied during the individual experiments (workload or task frequencies) is varied at a given step size. In all the tests, the smallest step size possible was used. Table 5.1 shows the parameter varied for each Hartstone experiment.

Table 5.1. Hartstone Experiments

Hartstone Experiment	Parameter Varied
1	Increase the frequency of the highest priority task
2	Increase the frequency of all tasks
3	Increase the workload of all tasks

5.5.1 Task Set A The purpose of this task set was to validate the behavior of RATESIM when executing periodic, harmonic user tasks. In all experiments (see Table 5.2) RATESIM successfully predicted the failure of the task set prior to actual failure. Note that all the predictions RATESIM made fell within the step interval that the Hartstone was able to achieve. Additionally, in all experiments, except experiment 1, the RATESIM and Hartstone failure modes were similar.

The RATESIM failure mode for experiment 1 was similar to Hartstone, but incomplete. It did not show any deadlines missed for the lowest priority task, Task 1. However, using the same task parameters that caused the Hartstone benchmark to fail, the failure mode is both similar and complete: Task 5 – 1 missed, Task 1 – 13 missed.

Table 5.2. Test Results - Task Set A - Periodic/Harmonic

Hartstone Experiment Number	Hartstone					RATESIM			
		Passed		Failed		Passed		Failed	
	Task	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets
1	Task 5	2500	2	2404	2	2458	2	2457	2
	Task 4	62500	4	62500	4	62500	4	62500	4
	Task 3	125000	8	125000	8	125000	8	125000	8
	Task 2	250000	16	250000	16	250000	16	250000	16
	Task 1	500000	32	500000	32	500000	32	500000	32
	Failure Mode (Deadlines): Hartstone – Task 1: 10 missed, 10 skipped; Task 5: 1 missed RATESIM – Task 5: 2 missed								
2	Task 5	8224	2	8013	2	8123	2	8122	2
	Task 4	16447	4	16026	4	16246	4	16244	4
	Task 3	32895	8	32051	8	32492	8	32488	8
	Task 2	65789	16	64103	16	64984	16	64976	16
	Task 1	131579	32	128205	32	129968	32	129952	32
	Failure Mode (Deadlines): Hartstone – Task 1: 1 met, 39 missed, 39 skipped RATESIM – Task 1: 1 missed								
3	Task 5	31250	17	31250	18	31250	17.90	31250	17.91
	Task 4	62500	19	62500	20	62500	19.90	62500	19.91
	Task 3	125000	23	125000	24	125000	23.90	125000	23.91
	Task 2	250000	31	250000	32	250000	31.90	250000	31.91
	Task 1	500000	47	500000	48	500000	47.90	500000	47.91
	Failure Mode (Deadlines): Hartstone – Task 1: 10 missed, 10 skipped RATESIM – Task 1: 18 missed								

5.5.2 Task Set B The purpose of this task set was to validate the behavior of RATESIM when executing periodic, non-harmonic user tasks. In experiments 1 and 3 (see Table 5.3) RATESIM successfully predicted the failure of the task set prior to actual failure. Again, note that the predictions RATESIM made fell within the step interval that the Hartstone was able to achieve. Additionally, in all experiments, the RATESIM and Hartstone failure modes were similar.

RATESIM was unsuccessful in predicting the failure of the user task set in experiment 2. Curiously, the Hartstone benchmark failed to execute a user task set whose rate of execution was *lower* than a user task set it *successfully* executed. The cause of this behavior is most likely due to the runtime optimization associated with task suspension discussed in Chapter IV, Section 3, Page 4-12. If the delay for Task X will expire within T_w units of time of the delay that previously expired for Task Y, then the penalty to respond to Task X's delay expiration is much less than if the delay expiration had occurred after T_w units of time. Figure 5.1 illustrates the window that exists in which the delay optimization would occur. Figure 5.2 shows relationship between the penalty for delay expirations that fall within the T_w window and those that fall outside it. As the number of tasks outside the window increases, the penalty increased linearly. The same is true for those tasks within the T_w window, but the rate of increase is significantly smaller. This being the case, the determining factor in whether or not the increased penalty will be paid shifts to whether or not delay expirations are "clustered" within a T_w window. Therefore, it is possible, just as was seen in Hartstone experiment 2, that a task set whose frequencies were lower than another task set would fail while the task set with higher frequencies would pass.

This would also explain why this "T_w effect" was not observed in the task sets with harmonic frequencies. Due to the very nature of the relationship between the task frequencies, delay expirations will fall within the T_w window. Further, any jitter within the expiration of the delay, given a large enough T_w, would not be sufficient to push a task's delay expiration outside that window. With non-harmonic tasks sets, however, the relationship between the frequencies may or may not

cause them to fall within a T_w window and even if they do, delay jitter could cause the window to be exceeded.

Assuming that the above hypothesis is true, obviously the T_w that is modeled within RATESIM is not equal to the T_w that exists within the kernel that the Hartstone is executing under. In order to increase the accuracy of RATESIM and to predict failures such as that experienced during experiment 2, the T_w effect within the the XD Ada kernel should be more accurately characterized and subsequently modeled within RATESIM. A more specialized test than that provided by the ACEC should be constructed and run on the XD Ada kernel to determine the precise point at which the optimization will and won't occur. The challenge for this test will be to characterize precisely the remaining DELAY jitter since this is presumably what causes the optimize/no optimize threshold to be crossed.

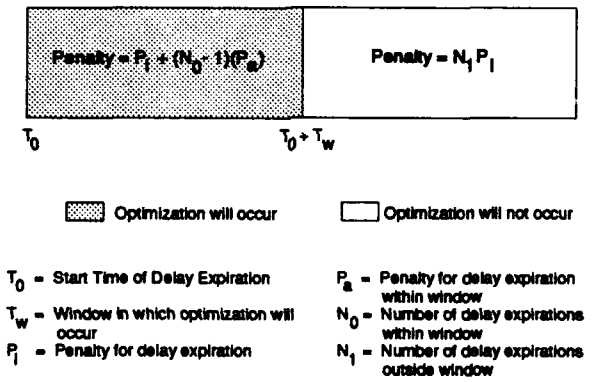

Figure 5.1. Delay Optimization

5.5.3 Task Set C The purpose of this task set was to validate the behavior of RATESIM when executing periodic, harmonic, dependent user tasks. In both experiments (see Tables 5.4 and 5.5) RATESIM successfully predicted the failure of the task set prior to actual failure. Note that, in this case, the predictions RATESIM made were more conservative than in task sets A and B. Even so, these results meet the criteria for successful validation of the model. Additionally, in both experiments, the RATESIM and Hartstone failure modes were similar.

5.6 Summary

In this chapter, the system task's behavior modeled in RATESIM was validated individually, that is, separate from the runtime system execution. From that perspective, it was shown that RATESIM was indeed modeling the runtime system behavior correctly. Then RATESIM was validated as a whole, that is, with all the system tasks interacting with each other as well as with the user task set. It was shown that RATESIM successfully met the objective of predicting the scheduling failure of each user task sets prior to the actual failure in every case except one. This failure, experiment 2 in the non-harmonic task set, was likely due to a runtime system optimization not sufficiently characterized by RATESIM. This hypothesis was discussed and a method for additional research was suggested.

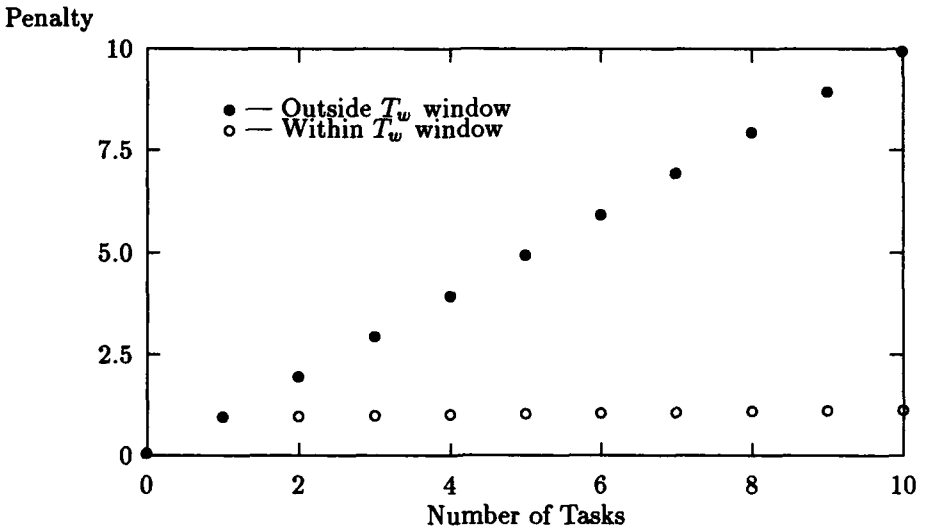

Figure 5.2. Delay Penalties

Table 5.3. Test Results - Task Set B - Periodic/Non-Harmonic

Hartstone Experiment Number	Task	Hartstone Passed Period (μs)	Hartstone Passed Kilo-Whets	Hartstone Failed Period (μs)	Hartstone Failed Kilo-Whets	RATESIM Passed Period (μs)	RATESIM Passed Kilo-Whets	RATESIM Failed Period (μs)	RATESIM Failed Kilo-Whets
1	Task 5	2488	2	2446	2	2488	2	2487	2
	Task 4	145138	4	145138	4	145138	4	145138	4
	Task 3	217865	8	217865	8	217865	8	217865	8
	Task 2	434783	16	434783	16	434783	16	434783	16
	Task 1	500000	32	500000	32	500000	32	500000	32
	Failure Mode (Deadlines): Hartstone – Task 5: 1 missed, 1 skipped RATESIM – Task 5: 3 missed								
2	Task 5	17551	2	17838	2	17135	2	17068	2
	Task 4	23409	4	23793	4	22852	4	22763	4
	Task 3	35139	8	35716	8	34305	8	34165	8
	Task 2	70126	16	71276	16	68446	16	68166	16
	Task 1	80645	32	81967	32	78740	32	78431	32
	Failure Mode (Deadlines): Hartstone – Task 1: 1 met, 39 missed, 39 skipped RATESIM – Task 1: 1 missed								
3	Task 5	108814	46	108814	47	108814	46.30	108814	46.40
	Task 4	145138	48	145138	49	145138	48.30	145138	48.40
	Task 3	217865	52	217865	53	217865	52.30	217865	52.40
	Task 2	434783	60	434783	61	434783	60.30	434783	60.40
	Task 1	500000	76	500000	77	500000	76.30	500000	76.40
	Failure Mode (Deadlines): Hartstone – Task 1: 9 missed, 9 skipped RATESIM – Task 1: 5 missed								

Table 5.4. Test Results - Task Set C1 - Periodic/Harmonic/Synchronization

Hartstone Experiment Number	Hartstone							
					Passed		Failed	
	Task	Entry Call at time (μs)	Accept at time (μs)	Accept KWhets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets
2	Task 5	0	n/a	n/a	9921	2	9827	2
	Task 4	n/a	0	0	9921	4	9827	4
	Task 3	n/a	n/a	n/a	39683	8	39308	8
	Task 2	n/a	n/a	n/a	79365	16	78616	16
	Task 1	n/a	n/a	n/a	158730	32	157233	32
	RATESIM							
					Passed		Failed	
	Task	Entry Call at time (μs)	Accept at time (μs)	Accept KWhets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets
	Task 5	0	n/a	n/a	10417	2	9921	2
	Task 4	n/a	0	0	10417	4	9921	4
	Task 3	n/a	n/a	n/a	41667	8	39683	8
	Task 2	n/a	n/a	n/a	83333	16	79365	16
	Task 1	n/a	n/a	n/a	166667	32	158730	32
	Failure Mode (Deadlines): Hartstone – Task 1: 32 missed, 32 skipped RATESIM – Task 1: 54 missed							

Table 5.5. Test Results - Task Set C2 - Periodic/Harmonic/Synchronization

Hartstone Experiment Number	Hartstone							
					Passed		Failed	
	Task	Entry Call at time (μs)	Accept at time (μs)	Accept KWhets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets
2	Task 5	0	n/a	n/a	8446	0	8224	0
	Task 4	n/a	0	2	8446	4	8224	4
	Task 3	n/a	n/a	n/a	33784	8	32895	8
	Task 2	n/a	n/a	n/a	67568	16	65789	16
	Task 1	n/a	n/a	n/a	135135	32	131579	32
	RATESIM							
					Passed		Failed	
	Task	Entry Call at time (μs)	Accept at time (μs)	Accept KWhets	Period (μs)	Kilo-Whets	Period (μs)	Kilo-Whets
	Task 5	0	n/a	n/a	8717	0	8705	0
	Task 4	n/a	0	2	8717	4	8705	4
	Task 3	n/a	n/a	n/a	34868	8	34819	8
	Task 2	n/a	n/a	n/a	69735	16	69638	16
	Task 1	n/a	n/a	n/a	139470	32	139276	32
	Failure Mode (Deadlines): Hartstone – Task 1: 38 missed, 38 skipped RATESIM – Task 1: 2 missed							

VI. Conclusions and Recommendations

6.1 Introduction

A fundamental problem in hard realtime systems is ensuring that tasks always meet their deadlines. The solution to this problem is to develop schedules which will ensure that result. However, since developing an optimal schedule can be very expensive in terms of time, various other scheduling techniques have been employed to bring the computational expense of developing a schedule down to a manageable level. One difficulty with these techniques, in the context of the overall system design, is that the system overhead associated with the execution of the user tasks is either underestimated, poorly understood, or simply not accounted for in the scheduling theory. So, in spite of the scheduling algorithm's prediction, task deadlines are missed.

This research effort's objective was three-fold:

1) To demonstrate that intimate knowledge of the entire runtime environment is not required to make an accurate determination of reserve capacity and schedulability – that is, a subset of key runtime parameters is sufficient.

2) To provide insight into a user task's interaction with the runtime environment during its execution.

3) To develop a parameterized model of a runtime environment which provides a conservative determination of task schedulability and processor reserve capacity.

All these objectives have been accomplished. The validation data in Chapter V demonstrated that relatively few runtime system parameters are needed to accurately predict the schedulability of the specified user tasks. Additionally, the measurement of these system parameters using software-based methods which depends on the resolution of the system clock does not adversely affect the accuracy of the conservative prediction. Insight into a user task's interaction with the runtime

system was provided through RATESIM's event history. This event history enables one to quickly determine the cause of a missed deadline without introduction of timing errors. Finally, RATESIM provided the parameterized model of a runtime environment to accomplish the last objective. Specifically, this research has accomplished the following:

i) Confirmed the hypothesis that a system task can indeed be modeled as a low priority user task that blocks the execution of a high priority user task. This means that a scheduling determination which includes any blocking a user task suffers from the runtime system, can be made by including runtime system blocking in the factor B_i in Equation 2.1, Chapter II, repeated below.

$$\forall i, 1 \leq i \leq n, min(\sum_{j=1}^{i-1} C_j \frac{1}{lT_k} \lceil \frac{lT_k}{T_j} \rceil + \frac{C_i}{lT_k} + \frac{B_i}{lT_k}) \leq 1$$
$$(k,l) \in R_i$$

where C_j and T_j are the execution time and period of task τ_j respectively, $R_i = \{(k, l) \mid 1 \leq k \leq i, l = 1, \ldots, \lfloor \frac{T_i}{T_k} \rfloor\}$ and B_i is the worst case blocking time for τ_i.

ii) Demonstrated that sufficient reserve capacity alone will not guarantee schedulability of a user task set. The frequency and phasing of system tasks, relative to the user task deadlines, is inexorably linked to the schedulability of the user task set.

iii) Demonstrated that the concept of using a generic, reusable, parameterized model of a runtime system is a viable and inexpensive alternative to the expensive "build it and see if it works" approach often used to build embedded systems.

iv) Provided the capability to determine the schedulability of a system, early in the design, without the use of the target hardware. This capability will permit system designers to perform a worst-case analysis on both the user task set and the runtime system. The model can be used to analyze the impact of user task set changes as well as runtime system changes.

v) Provided insight into how to build better and more powerful models/simulations of runtime environments.

vi) Demonstrated the benefits of doing detailed clock update analysis and associated delay modeling.

6.2 Conclusions

Several conclusions can be made as a result of this research into task schedulability. They are:

1) The requirements of a user task set and the performance of the runtime system must be considered and analyzed simultaneously! Failure to do this will directly impact the success of the design.

2) Runtime system optimizations can be extremely sensitive to the various relationships between task parameters. That is, small changes in user task requirements, either in workload, execution frequency, or synchronization, can render a previously schedulable task set unschedulable even though the additional requirements were modest!

3) System tasks (or system overhead) can be modeled within the existing framework of the rate monotonic scheduling theory. They are simply another source of blocking and can be modeled as such. No extensions to the theory is required. An execution budget, similar to that which is typically allocated to user tasks, should be maintained and tracked for system tasks as well.

6.3 Recommendations for Future Research

The RATESIM model is a proof of concept (or demonstration) of a more accurate way to predict the schedulability of a given user task set. However, RATESIM used only a subset of the tasking constructs that would be encountered in an actual embedded system. An expansion of this

subset of tasking constructs is necessary in the modeling and schedule prediction of more realistic systems. The following should be considered for inclusion in future versions of RATESIM:

i) synchronous and asynchronous I/O support,

ii) user task critical sections,

iii) aperiodic tasks,

iv) more robust synchronization support such as: timed entry calls and parameter passing during synchronization,

v) dynamic task creation and deletion,

vi) implementation of the priority ceiling protocol,

vii) and task communication protocols.

In addition, RATESIM validation was done on one particular target system: a MC68020 running under the XD Ada runtime environment. It is necessary, from the perspective of further validation, to perform similar validation tests on other targets and environments.

6.3.1 Runtime Environment Simulators From a broader perspective, much could be done to enhance the capability of simulations such as RATESIM through the creation of descriptive languages to describe the behavior of: (1) the user tasks, and (2) runtime environments (20). Such languages would provide a much richer characterization of the user tasks and runtime environments than that which is currently provided in RATESIM. In RATESIM, for instance, the runtime environment is defined by the user identifying the system tasks and supplying execution time parameters; the scheduling policy and other implicit behavior of the runtime environment is contained in the RATESIM program code. Suppose, however, that one would prefer to model a task set's behavior under a different scheduling policy such as earliest-deadline-first to determine the viability of that approach. Or perhaps, runtime environment X contains optimizations A, B,

and C while runtime environment Y has only B. Further, it is conceivable that an important system task might not be identified by the user and included in the simulation. These descriptive languages would allow such information to be easily extracted from the environment and exploited within the simulation to determine task set behavior under a wider variety of conditions.

Consider Figure 6.1. A benchmark such as the ACEC can be used to determine the timing behavior of a runtime environment (or it could be supplied by the compiler vendor). Additionally, a Runtime Environment Description Language (RDL) can be used to describe characteristics of the compiler and associated RTE such as: scheduling policy, runtime optimizations, conditions in which those optimization occur, and other information which would provide a complete characterization of the runtime environment. A Task Description Language (TDL) can supply similar information about the user tasks to execute under the runtime environment: execution times (worst, best, and average case), synchronization points, I/O requirements, critical sections, etc. Using this information, a simulator would construct a model of the environment and "execute" the task set. The results of the simulation would then be used to refine or modify the design. If the target hardware were available, the same TDL could be used to automatically construct a synthetic benchmark to execute on the hardware. This would be valuable for several purposes: (1) to evaluate potential targets in the absence of the actual application source code, (2) as a further validation of the RTE simulator results, (3) to allow the description of benchmarks which more accurately reflect the design rather than relying on widely used benchmarks which may or may not do so, and (4) the automatic benchmark generation capability would defeat those optimizations of compilers which exist only to boost the performance of the generated code under a recognized benchmark such as the Whetstone or the Dhrystone.

If the runtime environment was also being built by the system designers or not available, then design decisions could be specified in RDL and the impact on the user task set could be analyzed. This type of design and analysis tool would be invaluable to designers. The strength of

this approach is its generality and the ability to apply it early in the design phase. Creation and standardization of an RDL and TDL is critical to the success of this approach.

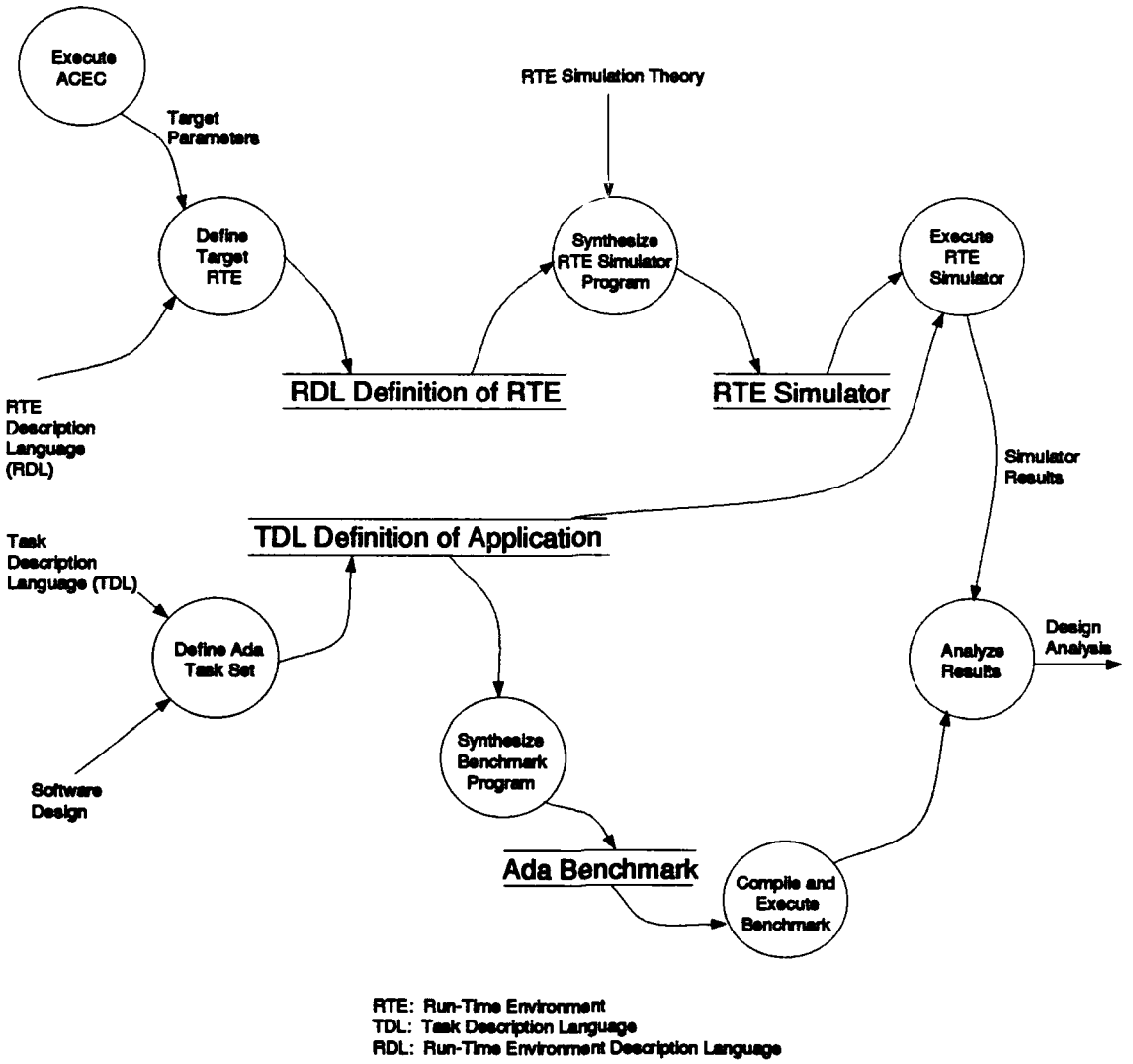

Figure 6.1. Recommendations for Future Research

Appendix A. *Delay Model*

A.1 XD Ada Delay

A.1.1 Overview The XD Ada MC68020 run-time system implements the Ada DELAY statement using two hardware timers on the MVME133A-20 Monoboard Microcomputer (14, 25). One of the timers ("timer A") is used as a continuously running "chime" clock and the other ("timer D") is used as an "alarm" clock to detect events that occur between chime clock updates. The delay model accounts for errors introduced into a given delay request due to:

1) Conversion from float to DURATION

2) Conversion from DURATION to SYSTEM.TICK

A.1.2 Hardware Timers Both timer A and timer D consist of an 8-bit counter and an associated prescaler value. The input frequency to the timers is $\frac{16}{13}MHz$. The pre-scaler value for both timers is 200 and so the resolution of the timers (and therefore SYSTEM.TICK) is:

$$200(\frac{1}{\frac{16}{13}MHz}) = 162.5\mu s$$

Elapsed time is maintained in a 64-bit register with the lower 8-bits being supplied by timer A. When the timer A counter overflows (every 0.0416s) an interrupt is generated and the most significant 56 bits of the 64-bit register are incremented. If a DELAY expires between timer A interrupts, timer D is set to interrupt within that interval.

A.2 Delay Model

The XD Ada DELAY implementation is modeled using the following equation:

$$ActualDelay = 162.5\mu s(floor(\frac{round(\frac{DelayRequest}{DURATION'SMALL})}{2.6624}) + 1) \qquad (A.1)$$

where Delay Request is the desired delay in seconds, DURATION'SMALL is 2^{-14} seconds, $162.5\mu s$ is SYSTEM.TICK, 2.6624 is the ratio $\frac{162.5\mu s}{2^{-14}_s}$ (dimensionless), and Actual Delay is the delay produced by timer A and timer D in seconds.

A.2.1 Error Sources There are two sources of error in the XD Ada MC68020 implementation of the DELAY statement. The first comes from the conversion from the requested delay to type DURATION. This conversion is performed by the function $round(\frac{DelayRequest}{DURATION'SMALL})$ in Equation A.1. The round function introduces a maximum of $\pm 0.5(DURATION'SMALL) = \pm 30.51757\mu s$ of error.

The second source of error is from the conversion of DURATION to SYSTEM.TICK. The conversion is performed by the function $floor() + 1$ in Equation A.1. The error will be at most SYSTEM.TICK seconds and at the least 0 seconds.

The worst case error ranges from $-30.51757\mu s$ less than the requested delay to $193.01757\mu s$ more than the request delay.

A.3 Sample Calculations

A.3.1 $DelayRequest = 460.0\mu s$ $ActualDelay = 162.5\mu s(floor(\frac{round(\frac{460s10^{-6}_s}{2^{-14}_s})}{2.6624}) + 1) = 650.0\mu s$

$AdditionalDelay = (650.0 - 460.0)\mu s = 190\mu s$

This sample calculation shows how a delay request of $460.0\mu s$ from a user application will produce a $650.0\mu s$ delay at the hardware timer level. An additional delay of $190\mu s$.

A.3.2 $DelayRequest = 10100.0\mu s$ $ActualDelay = 162.5\mu s(floor(\frac{round(\frac{10100s10^{-6}_s}{2^{-14}_s})}{2.6624}) + 1) = 10075.0\mu s$

$$AdditionalDelay = (10075.0 - 10100.0)\mu s = -25.0\mu s$$

This sample calculation shows how a delay request of $10100.0\mu s$ from a user application will produce a $10075.0\mu s$ delay at the hardware timer level. An additional delay of $-25\mu s$.

A.4 Statistics and Model Error

The data for the observed delay graphs was obtained from the ACEC benchmark test dt_dp_delay_01. The test originally tested only one delay request value and so was modified to test multiple delay requests at a specified step interval. The data contained in this appendix is from the modified test.

The descriptive statistics in Table A.1 are the additional delay observed in the delay graphs. Any data points that were less than or equal to 3.8 are not included in the descriptive statistics as those data points were actual delays.

Table A.1. Descriptive Statistics - Observed Additional Delay

Statistic	Value
Number of data points	1964
Lower 95% confidence interval	333.65
Mean	335.89
Upper 95% confidence interval	338.12
Standard deviation	50.483
Minimum value	208.40
1st Quartile	305.30
Median	332.40
3rd Quartile	370.37
Maximum value	446.80
Skew	-0.0555
Kurtosis	-0.7587

In order to account for the worst case additional delay in the RATESIM model, the maximum model error must be found. Figure A.1 is a graph of the model error versus the delay request. The model error is the difference between the observed additional delay and the model predicted additional delay. Table A.2 contain the descriptive statistics of the model error.

Table A.2. Descriptive Statistics - Model Error

Statistic	Value
Number of data points	1272
Lower 95% confidence interval	251.45
Mean	252.66
Upper 95% confidence interval	253.87
Standard deviation	22.044
Minimum value	187.20
1st Quartile	237.10
Median	256.3
3rd Quartile	270.00
Maximum value	304.40
Skew	-0.5547
Kurtosis	-0.2898

Recall that the delay model only accounts for the delay produced by the hardware timers. The observed delay contains, additionally, the context switch execution time, and a "delay execution component". The maximum error of the delay model is 305. The context switch as measured by the ACEC is a constant 149. In order for the delay model to always return the worst case additional delay, an additional 156 will be added to the delay model (305 - context switch = 156). Therefore, the worst case additional delay is modeled by adding an additional 305 to the existing delay model (Equation A.1).

Obviously, the data in Figure A.1 indicates that a systematic effect is still unaccounted for in the model error. The difficulty in determining what the effect is is that, although there is a pattern, there appears to be no correlation between the value and the original delay request. As shown in Figure A.2 for a given model prediction of additional delay the variation in actual delay ranges from 187.20 to 304.40.

One source of this variation may be due to the type of data collected for the model. Note the relatively constant difference between the observed and model additional delay's in the 1 μs data (Figure A.5). This relatively constant difference may also, in fact, hold in the delay requests shown in the $10, 100, \ldots, 10,000 \mu s$ data but is masked by the fact that the delay requests were issued in intervals greater than 1 μs. Therefore, the data obtained by using the greater intervals between

between delay requests may be various points along a line similar to the $1\mu s$ data. To determine if this is the case, more data should be taken and the increase in delay request value should be $1\mu s$. If this is the case, the delay model's delay execution component could be reduced further and a more accurate model would result.

Another possible source of this systematic error could lie in the *round* and *floor* functions used in the delay model. If a random sample of numbers in the range of 0 to 140,000 μs had the same distribution as that in Figure A.1, it would confirm this hypothesis.

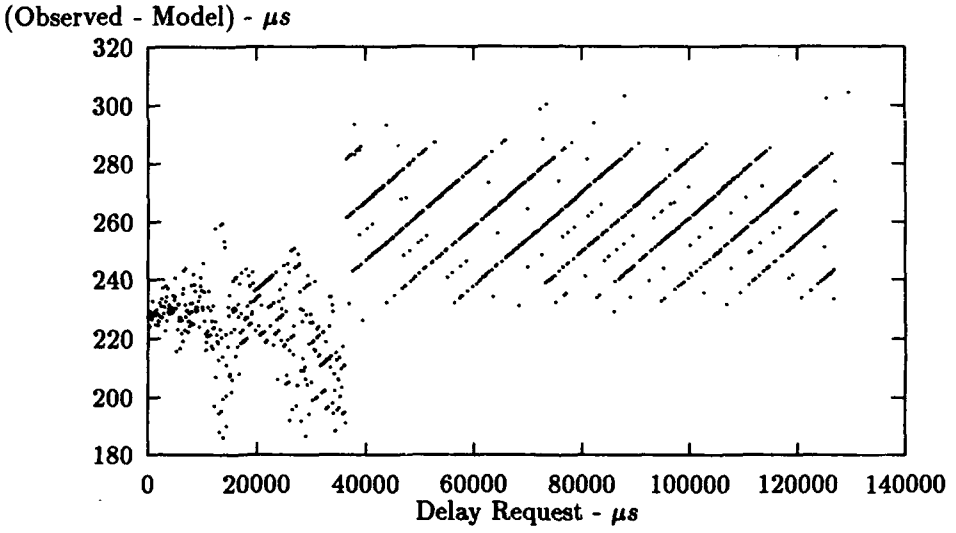

Figure A.1. (Observed Additional Delay - Model Additional Delay) vs. Delay Request

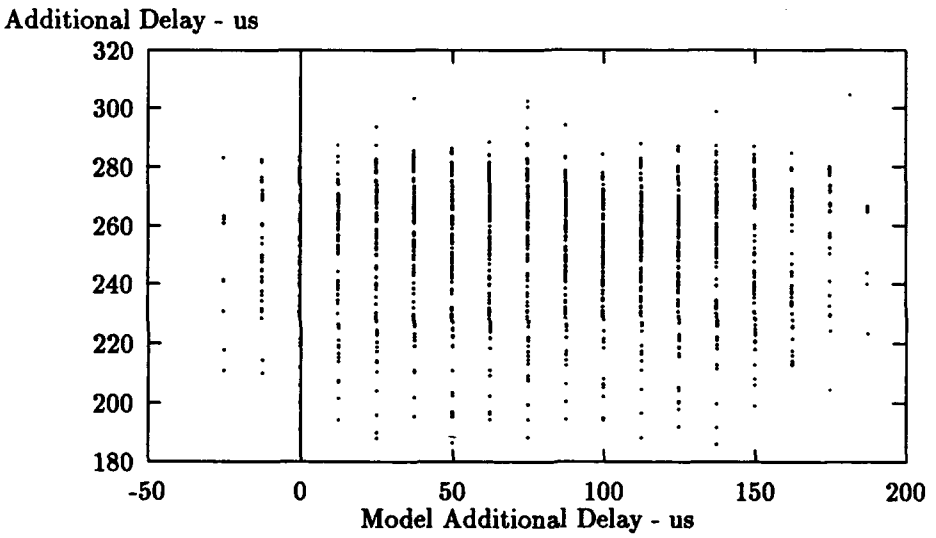

Figure A.2. Model Additional Delay vs. Observed Additional Delay - 100μs data

A.5 Delay Model and Observed Delay Graphs

The following section contains three types of graphs: (1) delay model, (2) observed delay, and (3) the combined delay model/observed delay. These graphs plot the requested delay versus the additional delay. Additional delay is defined as the delay experienced by the requesting task in addition to what was requested. The additional delay in the observed delay graphs contain three distinct components: (1) the delay produced by the hardware timers, (2) context switch execution time (assumed a constant $149\mu s$ as determined by the ACEC), and (3) a 'delay execution component' defined as the time remaining after (1) and (2) have been subtracted.

The delay model graphs, as shown, account for only component (1) above. The context switch execution time is accounted for (in the RATESIM model) by adding a constant to the result obtained from the delay model. The delay execution component is accounted for (in the RATESIM model) by adding another constant: the maximum value of (3) above. By adding the context switch execution time and the maximum delay execution component to the result obtained from the delay model, the result will always be greater than or equal to the maximum observed delay. This result is shown in the combined delay model/observed delay graphs.

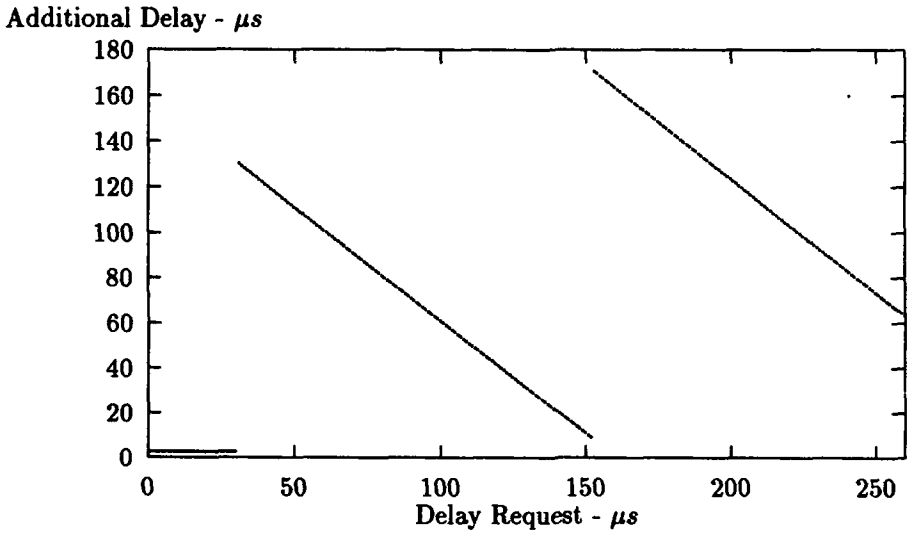

Figure A.3. Delay Model - 1μs

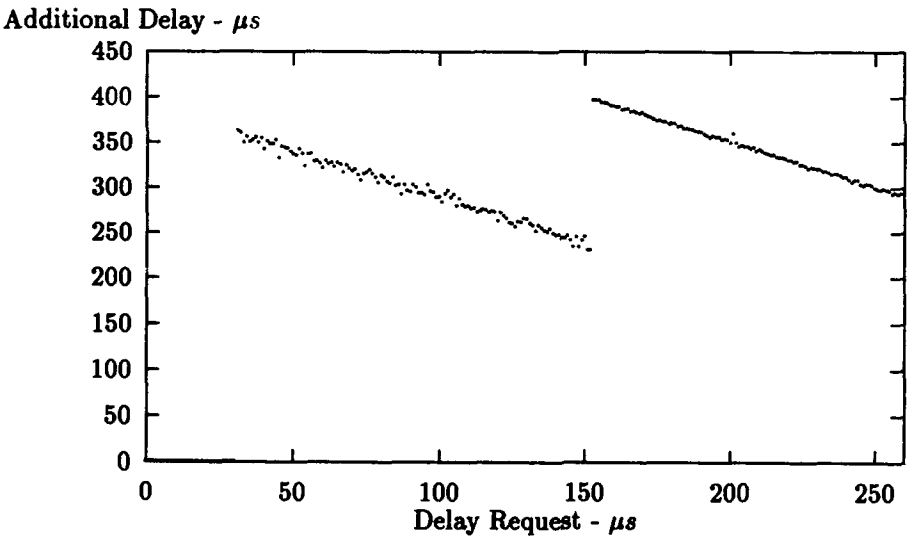

Figure A.4. Observed Delay - 1μs

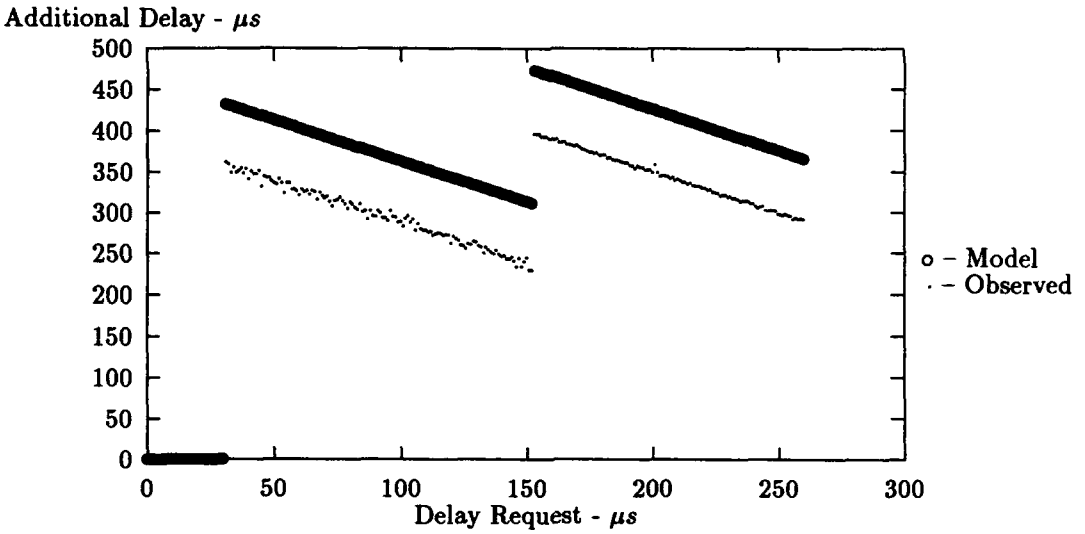

Figure A.5. Observed/Model Delay - $1\mu s$

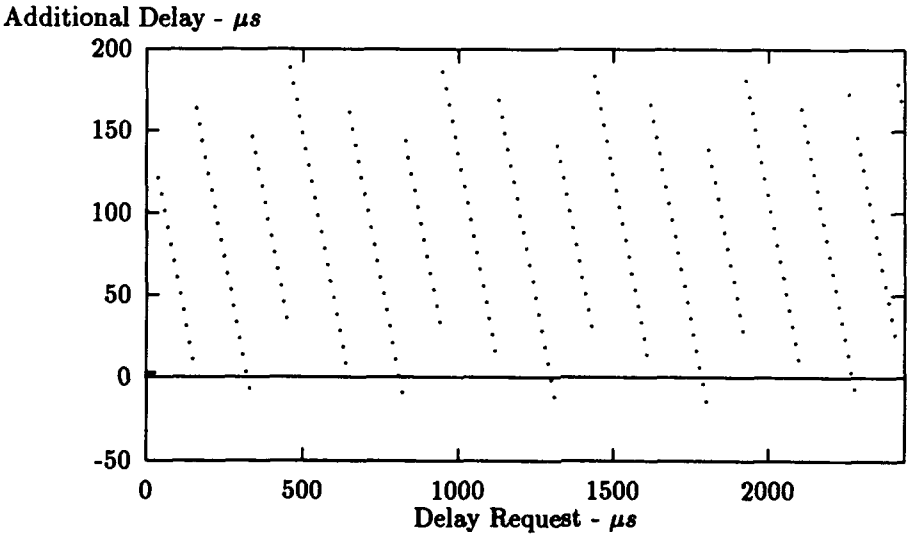

Figure A.6. Delay Model - $10\mu s$

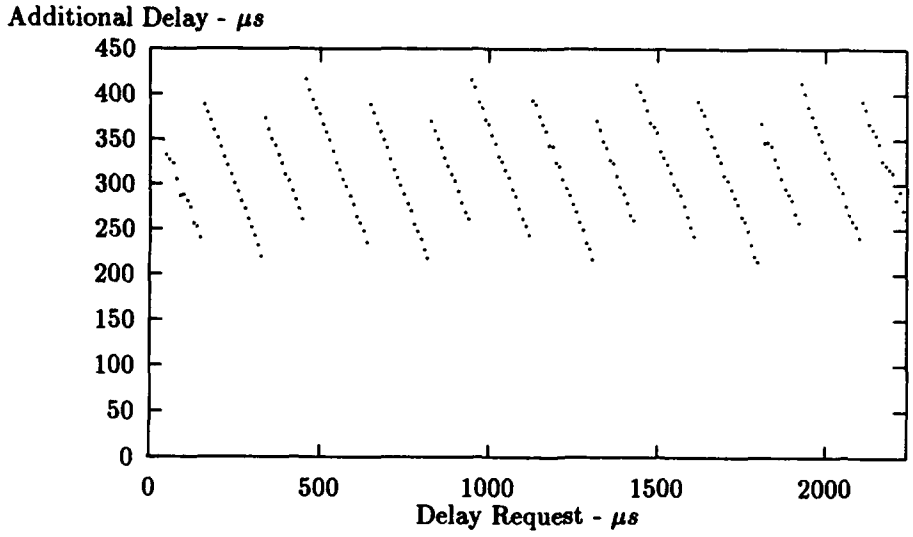

Figure A.7. Observed Delay - 10μs

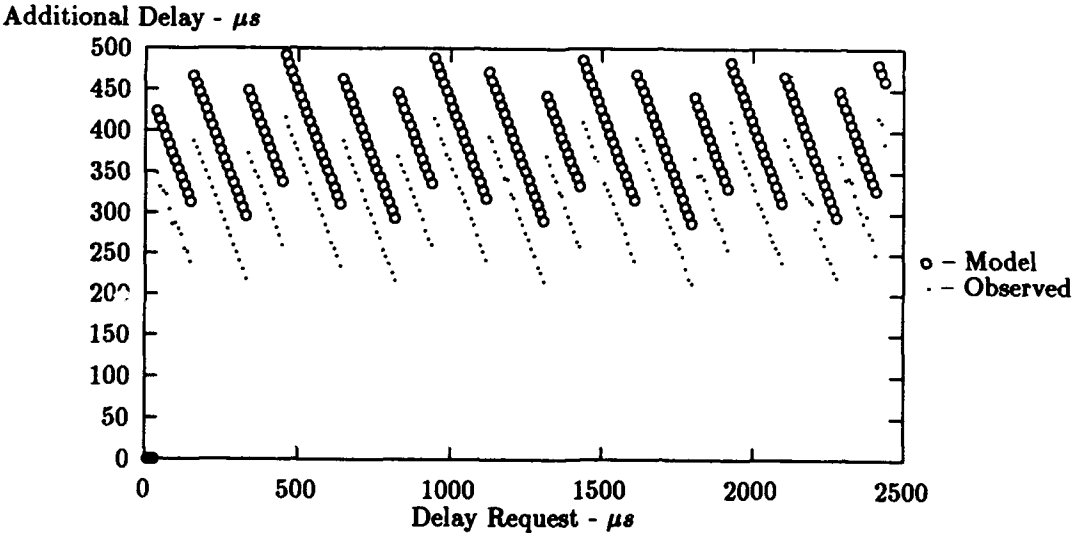

Figure A.8. Observed/Model Delay - 10μs

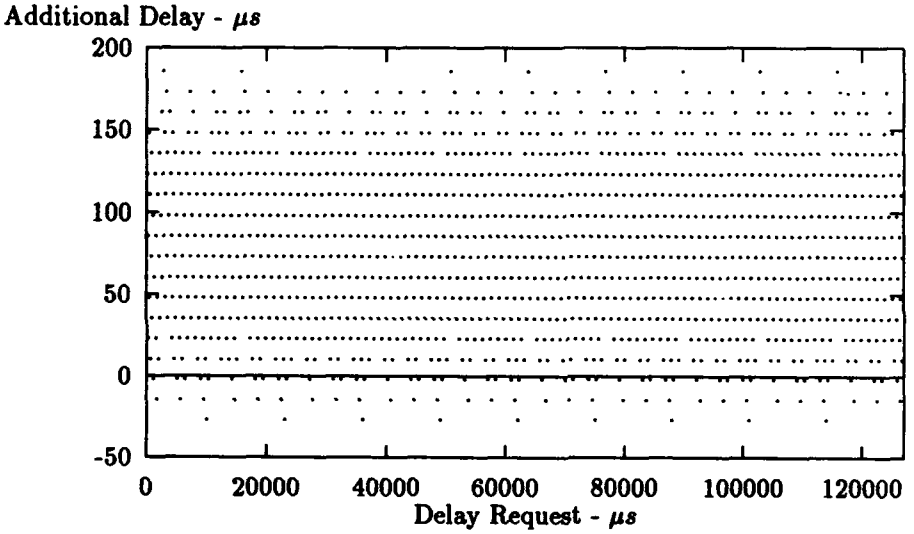

Figure A.9. Delay Model - 100μs

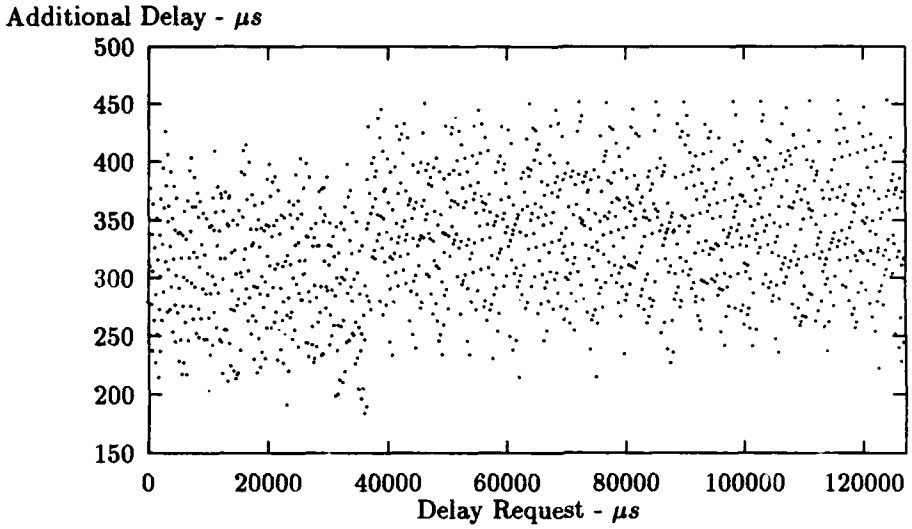

Figure A.10. Observed Delay - $100\mu s$

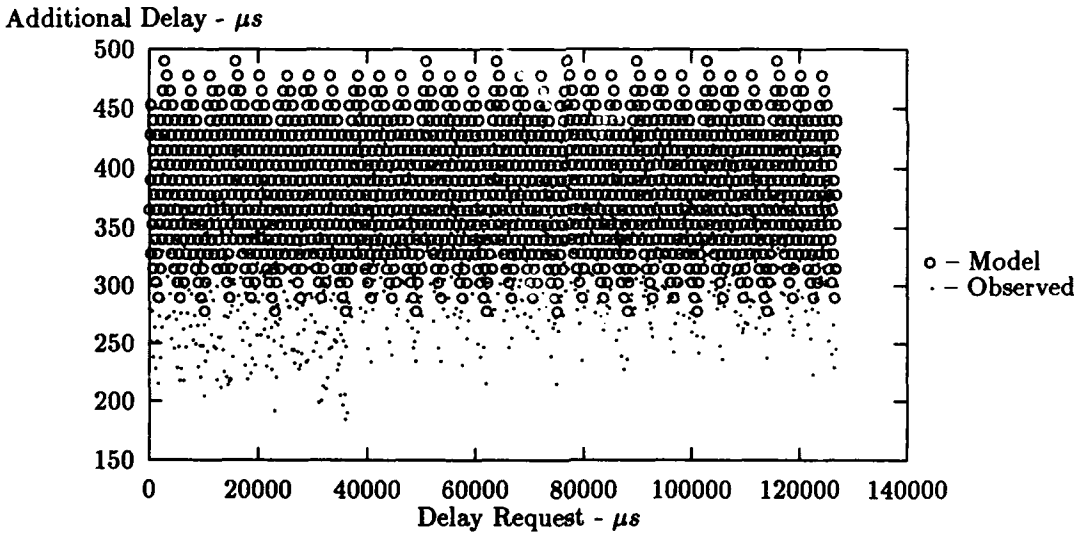

Figure A.11. Observed/Model Delay - $100\mu s$

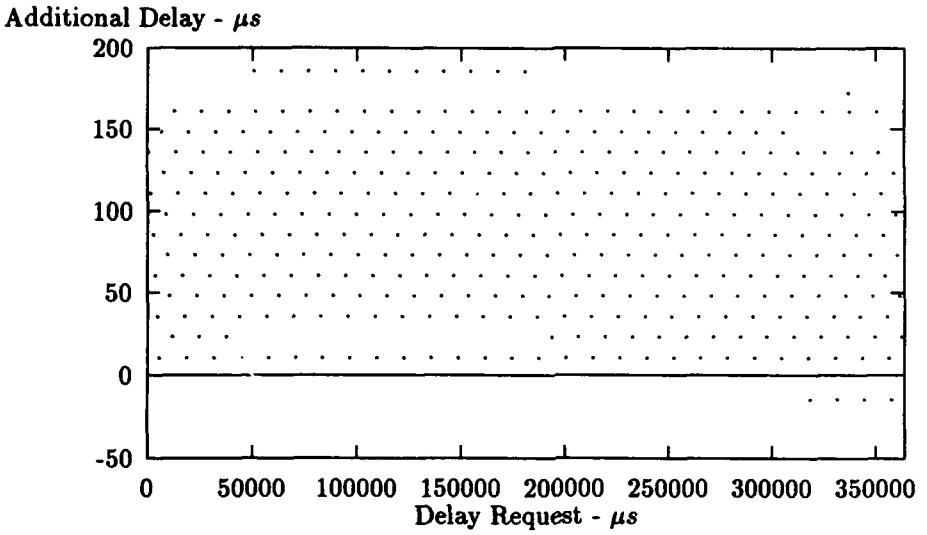

Figure A.12. Delay Model - 1000μs

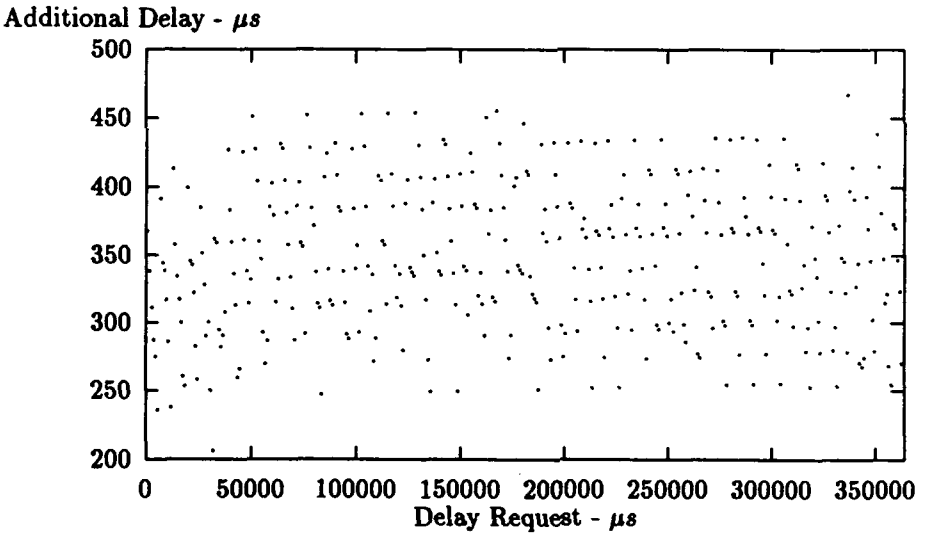

Figure A.13. Observed Delay - 1000μs

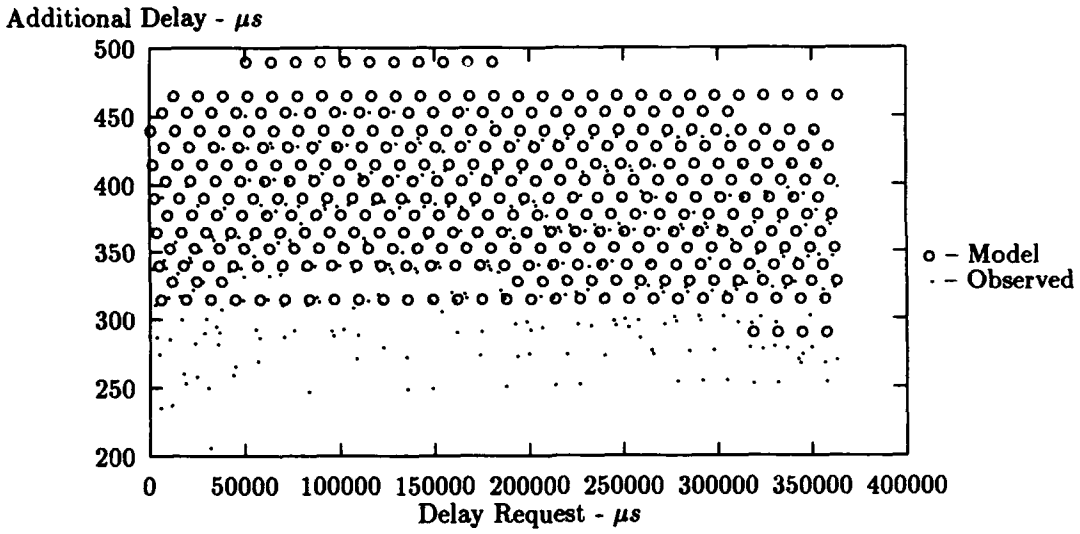

Figure A.14. Observed/Model Delay - 1000μs

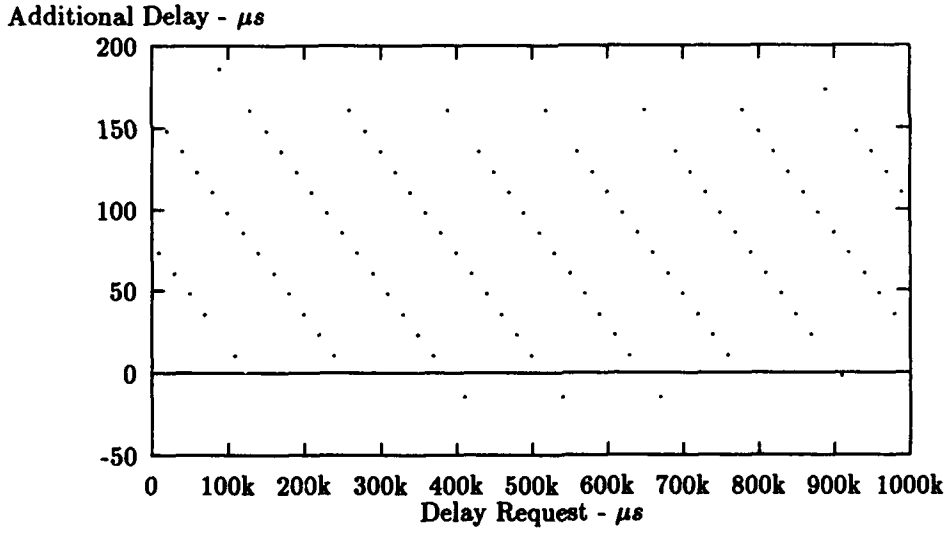

Figure A.15. Delay Model - 10000μs

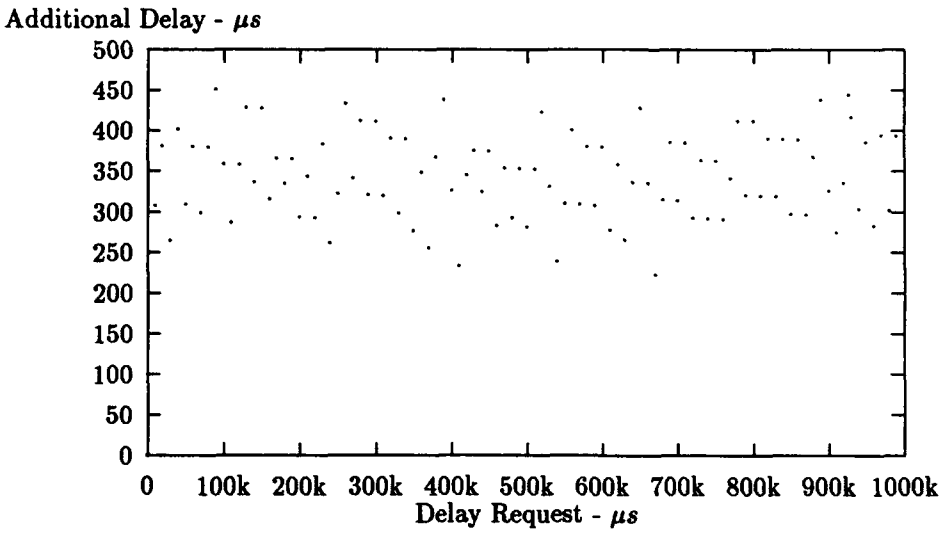

Figure A.16. Observed Delay - 10000μs

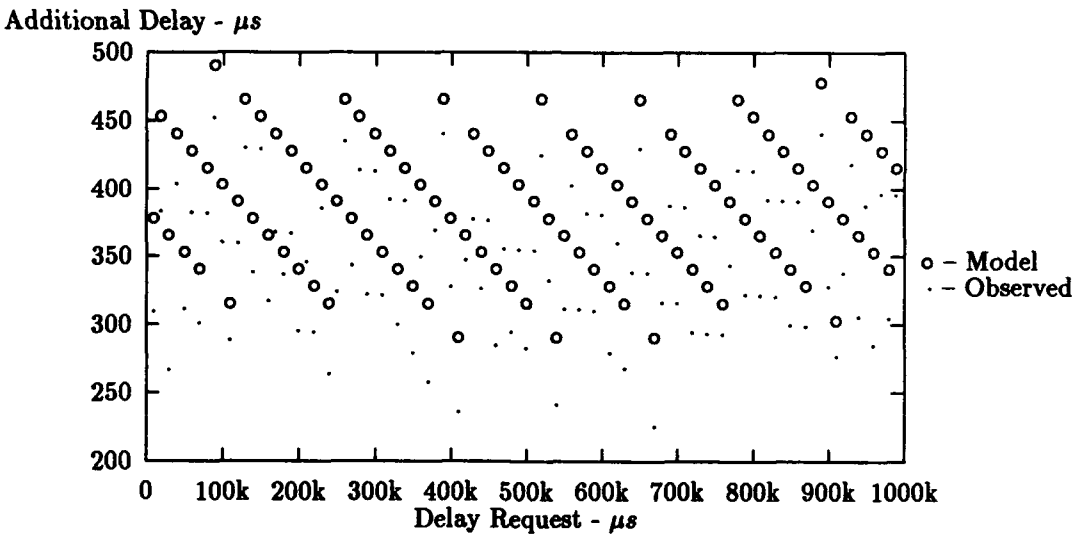

Figure A.17. Observed/Model Delay - 10000μs

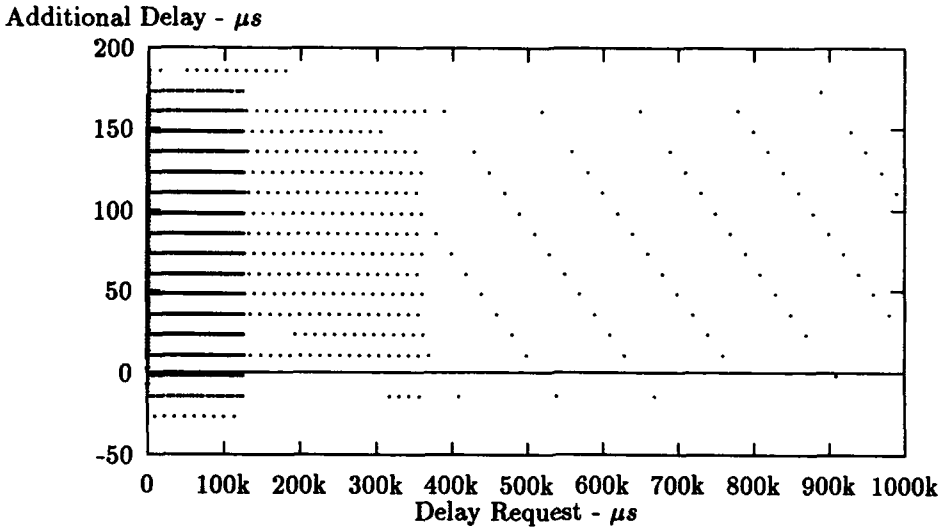

Figure A.18. Model Delay - All Data

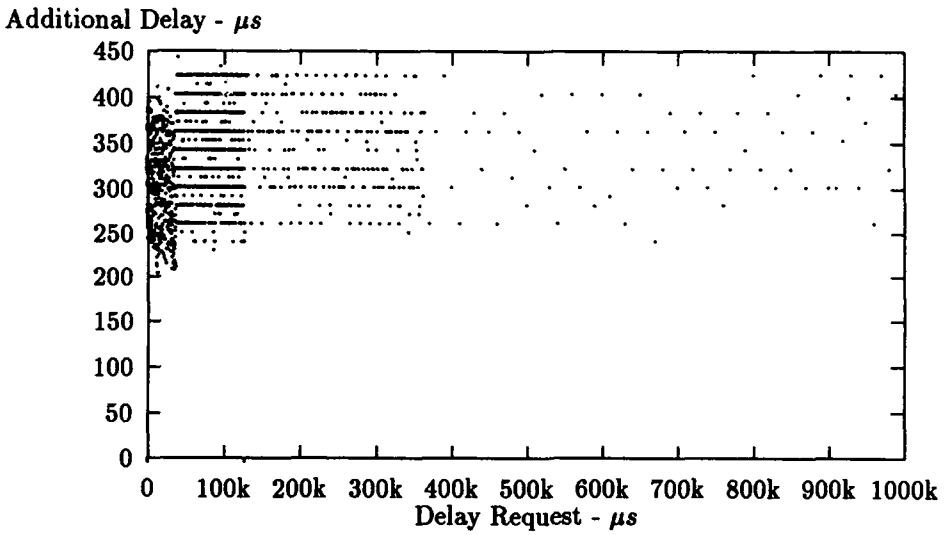

Figure A.19. Observed Delay - All Data

Table A.3. 1μs data

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
0	3.800*	25	3.800*	50	340.1	75	319.9
1	3.800*	26	3.800*	51	338.5	76	322.5
2	3.800*	27	3.800*	52	345.3	77	318.2
3	3.800*	28	3.800*	53	340.3	78	313.8
4	3.800*	29	3.800*	54	327.2	79	308.1
5	3.800*	30	3.800*	55	340.0	80	316.0
6	3.800*	31	365.8	56	340.5	81	314.1
7	3.800*	32	363.4	57	332.3	82	310.1
8	3.800*	33	352.6	58	332.5	83	307.9
9	3.800*	34	359.3	59	330.1	84	314.3
10	3.800*	35	353.3	60	324.8	85	304.6
11	3.800*	36	355.3	61	332.4	86	306.2
12	3.800*	37	358.7	62	329.0	87	295.8
13	3.800*	38	352.5	63	330.3	88	306.1
14	3.800*	39	357.5	64	326.3	89	299.7
15	3.800*	40	345.2	65	330.6	90	298.5
16	3.800*	41	354.4	66	328.7	91	306.0
17	3.800*	42	351.4	67	319.7	92	303.4
18	3.800*	43	351.5	68	328.0	93	297.0
19	3.800*	44	355.4	69	324.1	94	297.2
20	3.800*	45	335.8	70	321.0	95	294.5
21	3.800*	46	348.2	71	323.6	96	306.4
22	3.800*	47	346.8	72	317.2	97	298.7
23	3.800*	48	346.5	73	311.3	98	297.7
24	3.800*	49	342.8	74	318.3	99	291.8

* actual delay

Table A.4. 1μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
100	292.9	125	263.4	150	249.2	175	377.3
101	286.7	126	259.8	151	233.9	176	377.5
102	296.5	127	265.5	152	233.8	177	376.3
103	299.7	128	264.8	153	400.2	178	376.6
104	291.7	129	268.8	154	399.9	179	373.3
105	295.5	130	268.2	155	400.0	180	374.6
106	282.3	131	263.1	156	397.0	181	374.5
107	290.4	132	261.2	157	397.0	182	370.6
108	284.1	133	254.5	158	396.4	183	371.6
109	281.9	134	261.2	159	394.4	184	368.5
110	282.4	135	258.5	160	393.2	185	368.4
111	279.6	136	255.3	161	392.8	186	367.9
112	280.8	137	253.8	162	393.1	187	366.8
113	275.7	138	257.8	163	389.3	188	365.4
114	277.1	139	252.4	164	389.6	189	365.0
115	279.2	140	250.2	165	390.4	190	365.2
116	278.1	141	250.9	166	387.4	191	362.0
117	275.5	142	247.1	167	387.5	192	360.8
118	276.6	143	247.1	168	386.1	193	358.1
119	276.4	144	249.6	169	384.8	194	360.7
120	266.3	145	244.7	170	386.2	195	359.1
121	275.1	146	237.6	171	383.6	196	359.2
122	272.1	147	248.0	172	381.9	197	356.0
123	269.1	148	237.7	173	380.8	198	357.1
124	263.7	149	244.4	174	378.9	199	356.4

Table A.5. $1\mu s$ data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
200	352.8	225	328.5	250	302.4
201	362.4	226	327.3	251	301.3
202	352.7	227	324.4	252	300.3
203	348.5	228	325.2	253	301.7
204	349.7	229	325.0	254	300.8
205	348.7	230	323.1	255	299.9
206	348.3	231	323.7	256	296.0
207	346.1	232	320.9	257	297.3
208	347.7	233	321.4	258	296.0
209	343.3	234	319.5	259	296.8
210	345.3	235	317.3	260	295.9
211	343.6	236	318.3		
212	341.0	237	318.1		
213	340.8	238	316.8		
214	339.0	239	315.0		
215	340.0	240	315.0		
216	337.1	241	312.0		
217	336.6	242	309.1		
218	336.2	243	310.9		
219	335.3	244	311.5		
220	333.5	245	306.9		
221	333.6	246	305.8		
222	331.5	247	306.6		
223	331.9	248	307.2		
224	328.7	249	305.7		

Table A.6. 10μs data

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
0	3.800*	250	303.9	500	381.0	750	292.1
10	3.800*	260	294.5	510	369.7	760	281.0
20	3.800*	270	283.9	520	361.5	770	273.3
30	3.800*	280	275.1	530	352.7	780	258.3
40	352.5	290	263.9	540	339.6	790	250.4
50	335.7	300	255.0	550	326.7	800	242.0
60	330.2	310	245.1	560	319.4	810	230.2
70	326.2	320	234.6	570	308.6	820	220.8
80	308.0	330	222.8	580	300.2	830	373.2
90	289.4	340	376.1	590	289.7	840	362.6
100	291.3	350	364.5	600	280.7	850	353.6
110	284.0	360	353.5	610	267.0	860	343.9
120	277.0	370	346.4	620	259.9	870	332.9
130	259.1	380	335.2	630	250.8	880	323.2
140	255.4	390	326.5	640	238.4	890	313.9
150	243.2	400	314.2	650	391.7	900	306.9
160	391.1	410	307.0	660	383.1	910	295.8
170	383.0	420	295.5	670	371.5	920	282.3
180	374.5	430	285.7	680	362.1	930	271.6
190	363.7	440	275.6	690	352.8	940	264.9
200	354.6	450	264.1	700	343.9	950	419.3
210	345.0	460	419.7	710	331.3	960	410.6
220	333.5	470	407.9	720	319.4	970	394.4
230	324.0	480	397.1	730	310.2	980	388.0
240	313.8	490	387.6	740	301.8	990	375.0

* actual delay

Table A.7. 10μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
1000	369.6	1250	280.1	1500	360.6	1750	266.5
1010	358.0	1260	271.8	1510	340.1	1760	261.7
1020	348.7	1270	260.5	1520	332.4	1770	251.1
1030	334.4	1280	252.1	1530	325.4	1780	236.4
1040	328.6	1290	237.8	1540	315.9	1790	223.5
1050	319.0	1300	231.1	1550	303.4	1800	217.5
1060	310.3	1310	219.8	1560	297.5	1810	371.1
1070	297.3	1320	373.5	1570	291.2	1820	350.0
1080	289.6	1330	363.2	1580	278.9	1830	350.3
1090	276.3	1340	350.2	1590	266.9	1840	346.1
1100	264.4	1350	343.4	1600	255.3	1850	332.2
1110	257.0	1360	329.9	1610	245.3	1860	323.3
1120	246.7	1370	325.9	1620	395.4	1870	310.1
1130	396.6	1380	311.6	1630	387.4	1880	298.2
1140	391.7	1390	299.8	1640	380.4	1890	292.2
1150	379.5	1400	292.0	1650	366.4	1900	285.5
1160	370.1	1410	281.6	1660	357.8	1910	270.3
1170	361.7	1420	268.3	1670	345.4	1920	260.1
1180	346.5	1430	262.9	1680	336.4	1930	415.9
1190	345.0	1440	414.0	1690	328.6	1940	403.9
1200	327.2	1450	406.3	1700	313.5	1950	389.4
1210	322.7	1460	397.0	1710	307.4	1960	379.3
1220	308.7	1470	385.9	1720	297.4	1970	368.7
1230	298.1	1480	372.3	1730	288.1	1980	360.9
1240	290.4	1490	367.6	1740	281.0	1990	352.9

Table A.8. $10\mu s$ data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
2000	340.0	2250	250.7
2010	333.3	2260	244.7
2020	316.6	2270	233.1
2030	311.0	2280	225.4
2040	301.6	2290	374.8
2050	293.9	2300	365.5
2060	278.7	2310	345.1
2070	270.0	2320	347.5
2080	262.7	2330	343.5
2090	256.2	2340	327.4
2100	243.9	2350	313.4
2110	395.4	2360	306.1
2120	387.3	2370	292.6
2130	371.3	2380	301.4
2140	364.8	2390	280.7
2150	358.1	2400	274.3
2160	349.4	2410	254.1
2170	329.9	2420	420.9
2180	324.7	2430	415.2
2190	320.7	2440	389.5
2200	316.9		
2210	285.9		
2220	295.4		
2230	274.7		
2240	265.4		

Table A.9. 100μs data

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
100	282.1	2600	397.0	5100	341.6	7600	267.2
200	353.1	2700	290.9	5200	231.5	7700	325.4
300	253.6	2800	363.3	5300	293.9	7800	389.5
400	312.9	2900	428.7	5400	341.9	7900	295.5
500	379.7	3000	317.2	5500	249.4	8000	364.2
600	280.2	3100	381.8	5600	320.2	8100	255.2
700	340.5	3200	273.8	5700	220.2	8200	320.6
800	240.2	3300	351.4	5800	278.3	8300	376.1
900	308.8	3400	408.7	5900	339.6	8400	276.4
1000	366.4	3500	310.1	6000	230.4	8500	355.5
1100	266.4	3600	364.7	6100	303.4	8600	251.9
1200	329.1	3700	275.6	6200	360.4	8700	312.3
1300	230.4	3800	330.1	6300	264.8	8800	368.8
1400	288.6	3900	393.7	6400	336.3	8900	267.7
1500	354.3	4000	295.0	6500	220.2	9000	331.9
1600	254.3	4100	354.4	6600	300.7	9100	231.9
1700	318.4	4200	256.4	6700	369.0	9200	293.3
1800	217.1	4300	319.2	6800	249.0	9300	358.8
1900	275.6	4400	381.4	6900	320.0	9400	250.1
2000	339.6	4500	279.9	7000	395.0	9500	322.9
2100	239.4	4600	344.0	7100	278.5	9600	222.9
2200	303.9	4700	248.1	7200	343.0	9700	288.9
2300	370.8	4800	305.3	7300	405.8	9800	354.3
2400	266.1	4900	371.5	7400	298.5	9900	250.0
2500	332.1	5000	272.8	7500	364.4	10000	306.9

Table A.10. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
10100	206.9	12600	301.9	15100	221.3	17600	332.9
10200	268.4	12700	360.1	15200	284.4	17700	232.9
10300	346.8	12800	257.8	15300	361.0	17800	295.6
10400	237.0	12900	319.6	15400	240.7	17900	365.2
10500	289.7	13000	376.6	15500	307.3	18000	265.8
10600	347.9	13100	258.1	15600	393.6	18100	344.7
10700	245.0	13200	349.3	15700	303.3	18200	221.2
10800	304.8	13300	214.1	15800	355.4	18300	294.6
10900	372.1	13400	347.6	15900	412.1	18400	353.1
11000	273.8	13500	371.6	16000	319.2	18500	250.6
11100	344.4	13600	246.4	16100	391.6	18600	329.8
11200	412.2	13700	373.1	16200	280.9	18700	226.2
11300	297.2	13800	227.6	16300	327.9	18800	289.0
11400	359.9	13900	275.8	16400	417.1	18900	345.6
11500	260.0	14000	324.7	16500	293.6	19000	238.8
11600	330.7	14100	291.9	16600	370.2	19100	304.2
11700	381.2	14200	352.4	16700	266.0	19200	367.6
11800	289.7	14300	222.7	16800	309.4	19300	277.3
11900	342.9	14400	274.8	16900	402.2	19400	343.6
12000	249.8	14500	326.9	17000	285.7	19500	233.6
12100	319.7	14600	216.2	17100	344.6	19600	299.6
12200	375.0	14700	306.1	17200	254.6	19700	348.8
12300	248.2	14800	219.3	17300	330.9	19800	259.4
12400	371.4	14900	278.9	17400	374.0	19900	322.2
12500	220.7	15000	348.4	17500	270.1	20000	387.8

Table A.11. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
20100	287.8	22600	229.3	25100	330.4	27600	230.3
20200	347.7	22700	282.3	25200	369.1	27700	337.3
20300	400.5	22800	354.8	25300	289.4	27800	233.7
20400	313.4	22900	254.8	25400	318.3	27900	296.5
20500	373.2	23000	317.5	25500	405.2	28000	349.5
20600	259.7	23100	194.0	25600	283.9	28100	243.7
20700	338.9	23200	270.6	25700	358.3	28200	285.5
20800	385.2	23300	322.7	25800	284.1	28300	330.5
20900	294.9	23400	222.7	25900	306.5	28400	264.4
21000	347.9	23500	285.5	26000	379.6	28500	330.7
21100	264.4	23600	368.6	26100	306.1	28600	377.0
21200	310.7	23700	268.6	26200	318.2	28700	273.1
21300	389.9	23800	314.9	26300	246.6	28800	363.3
21400	273.5	23900	357.3	26400	338.4	28900	236.5
21500	352.7	24000	273.8	26500	401.2	29000	295.1
21600	232.3	24100	346.2	26600	246.9	29100	379.2
21700	315.5	24200	250.1	26700	337.8	29200	237.5
21800	378.2	24300	303.2	26800	234.9	29300	351.6
21900	257.9	24400	365.9	26900	290.9	29400	380.5
22000	341.0	24500	258.1	27000	389.4	29500	317.6
22100	241.0	24600	331.2	27100	282.7	29600	356.8
22200	294.1	24700	391.4	27200	345.4	29700	277.1
22300	356.8	24800	297.2	27300	245.4	29800	295.3
22400	266.5	24900	354.2	27400	260.7	29900	389.1
22500	312.8	25000	243.5	27500	330.3	30000	292.0

Table A.12. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
30100	331.5	32600	266.1	35100	207.2	37600	364.1
30200	245.1	32700	345.0	35200	260.0	37700	406.5
30300	314.3	32800	222.1	35300	326.3	37800	326.8
30400	350.3	32900	301.3	35400	239.9	37900	389.6
30500	270.6	33000	364.1	35500	289.1	38000	319.6
30600	330.1	33100	247.6	35600	198.7	38100	371.7
30700	253.1	33200	310.4	35700	254.7	38200	394.8
30800	292.2	33300	400.3	35800	331.0	38300	335.4
30900	338.5	33400	289.6	35900	207.8	38400	357.5
31000	262.1	33500	372.7	36000	270.6	38500	440.6
31100	321.6	33600	249.2	36100	187.0	38600	340.6
31200	201.3	33700	335.5	36200	249.8	38700	423.7
31300	287.6	33800	377.9	36300	296.1	38800	283.0
31400	347.2	33900	295.0	36400	192.2	38900	356.5
31500	240.4	34000	324.2	36500	345.1	39000	448.3
31600	303.1	34100	255.2	36600	387.5	39100	328.9
31700	203.1	34200	283.1	36700	287.5	39200	412.0
31800	262.0	34300	359.4	36800	350.3	39300	271.3
31900	324.8	34400	253.6	36900	433.4	39400	354.4
32000	215.1	34500	301.8	37000	283.0	39500	377.5
32100	298.2	34600	228.9	37100	375.8	39600	317.2
32200	350.3	34700	264.6	37200	275.8	39700	379.9
32300	261.0	34800	351.6	37300	358.9	39800	279.9
32400	303.4	34900	234.1	37400	401.3	39900	342.7
32500	213.1	35000	307.6	37500	321.6	40000	405.5

Table A.13. $100\mu s$ data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
40100	305.5	42600	409.6	45100	330.7	47600	272.1
40200	368.2	42700	309.6	45200	393.5	47700	334.9
40300	247.9	42800	352.0	45300	313.8	47800	397.6
40400	321.3	42900	435.2	45400	335.9	47900	318.0
40500	393.7	43000	335.2	45500	276.6	48000	360.4
40600	293.7	43100	377.6	45600	339.3	48100	280.7
40700	336.2	43200	297.9	45700	402.1	48200	313.5
40800	236.2	43300	340.3	45800	281.7	48300	406.2
40900	319.3	43400	423.4	45900	364.8	48400	265.6
41000	361.7	43500	323.4	46000	427.6	48500	348.7
41100	282.0	43600	365.8	46100	337.3	48600	248.7
41200	324.4	43700	265.8	46200	370.0	48700	311.4
41300	397.9	43800	369.3	46300	453.1	48800	394.5
41400	307.5	43900	371.0	46400	353.1	48900	253.8
41500	350.0	44000	291.4	46500	395.5	49000	357.3
41600	433.1	44100	354.1	46600	306.2	49100	236.9
41700	333.1	44200	254.1	46700	358.3	49200	299.7
41800	375.5	44300	337.2	46800	400.7	49300	342.1
41900	295.8	44400	400.0	46900	341.4	49400	262.5
42000	338.2	44500	300.0	47000	383.8	49500	325.2
42100	421.4	44600	362.8	47100	304.2	49600	388.0
42200	321.4	44700	242.4	47200	336.9	49700	288.0
42300	384.1	44800	305.2	47300	389.0	49800	341.1
42400	426.5	44900	367.9	47400	289.0	49900	433.9
42500	346.9	45000	288.3	47500	381.8	50000	293.2

Table A.14. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
50100	376.3	52600	297.3	55100	401.5	57600	322.6
50200	276.3	52700	380.4	55200	291.8	57700	233.2
50300	339.0	52800	300.8	55300	384.6	57800	305.7
50400	392.1	52900	363.5	55400	447.4	57900	348.1
50500	322.1	53000	406.0	55500	347.4	58000	288.8
50600	364.5	53100	306.0	55600	410.1	58100	351.5
50700	427.3	53200	368.7	55700	310.1	58200	393.9
50800	347.7	53300	248.4	55800	372.9	58300	293.9
50900	369.7	53400	331.5	55900	435.6	58400	356.7
51000	432.5	53500	394.2	56000	335.6	58500	256.7
51100	332.5	53600	294.2	56100	368.4	58600	319.5
51200	395.2	53700	336.6	56200	278.1	58700	402.6
51300	335.9	53800	236.6	56300	361.2	58800	272.5
51400	368.7	53900	319.8	56400	423.9	58900	324.6
51500	441.1	54000	382.5	56500	283.2	59000	407.7
51600	341.1	54100	282.5	56600	366.3	59100	328.1
51700	403.9	54200	324.9	56700	266.3	59200	370.5
51800	303.9	54300	408.0	56800	349.4	59300	433.3
51900	366.6	54400	308.0	56900	412.2	59400	353.6
52000	429.4	54500	370.8	57000	312.2	59500	396.0
52100	329.4	54600	433.6	57100	375.0	59600	296.0
52200	392.2	54700	333.6	57200	275.0	59700	358.8
52300	271.8	54800	376.0	57300	337.7	59800	401.2
52400	334.6	54900	276.0	57400	359.8	59900	341.9
52500	417.7	55000	359.1	57500	280.1	60000	384.3

Table A.15. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
60100	304.7	62600	388.5	65100	329.9	67600	434.0
60200	347.1	62700	288.5	65200	413.0	67700	334.0
60300	430.2	62800	361.9	65300	272.3	67800	376.5
60400	330.2	62900	393.7	65400	355.4	67900	276.5
60500	372.6	63000	314.0	65500	418.2	68000	359.6
60600	252.2	63100	376.8	65600	318.2	68100	402.0
60700	335.3	63200	276.8	65700	401.3	68200	302.0
60800	418.5	63300	339.5	65800	280.9	68300	385.1
60900	318.5	63400	422.6	65900	364.0	68400	447.9
61000	340.5	63500	281.9	66000	386.1	68500	307.2
61100	240.5	63600	365.0	66100	306.4	68600	390.3
61200	344.0	63700	397.8	66200	348.9	68700	290.3
61300	366.0	63800	327.8	66300	248.9	68800	373.4
61400	306.7	63900	390.6	66400	332.0	68900	436.1
61500	349.1	64000	453.3	66500	374.4	69000	336.1
61600	249.1	64100	353.3	66600	294.7	69100	378.5
61700	311.9	64200	395.7	66700	357.5	69200	278.5
61800	354.3	64300	295.7	66800	257.5	69300	341.3
61900	295.0	64400	358.5	66900	299.9	69400	424.4
62000	357.8	64500	431.9	67000	383.0	69500	324.4
62100	217.1	64600	341.6	67100	283.0	69600	387.2
62200	300.2	64700	404.4	67200	325.4	69700	266.8
62300	342.6	64800	304.4	67300	408.5	69800	329.6
62400	283.3	64900	367.1	67400	308.5	69900	392.3
62500	305.4	65000	429.9	67500	350.9	70000	303.0

Table A.16. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
70100	345.4	72600	296.5	75100	217.6	77600	342.1
70200	275.5	72700	349.6	75200	300.7	77700	384.5
70300	317.9	72800	442.4	75300	333.4	77800	304.8
70400	401.0	72900	352.1	75400	263.4	77900	367.6
70500	301.0	73000	384.8	75500	326.2	78000	430.4
70600	363.7	73100	284.8	75600	389.0	78100	330.4
70700	263.7	73200	347.6	75700	289.0	78200	393.1
70800	306.1	73300	390.0	75800	351.7	78300	313.5
70900	389.3	73400	330.7	75900	425.1	78400	355.9
71000	289.3	73500	393.4	76000	294.1	78500	418.6
71100	331.7	73600	273.1	76100	377.2	78600	298.3
71200	394.4	73700	376.5	76200	277.2	78700	371.7
71300	314.8	73800	378.3	76300	340.0	78800	281.4
71400	357.2	73900	318.9	76400	402.8	78900	344.2
71500	257.2	74000	361.4	76500	323.1	79000	406.9
71600	320.0	74100	241.0	76600	355.8	79100	306.9
71700	382.7	74200	344.5	76700	407.9	79200	369.7
71800	282.7	74300	386.9	76800	298.3	79300	269.7
71900	365.8	74400	266.5	76900	411.4	79400	312.1
72000	428.6	74500	329.3	77000	453.8	79500	395.2
72100	308.2	74600	270.0	77100	353.8	79600	295.2
72200	391.3	74700	312.4	77200	386.5	79700	337.6
72300	454.1	74800	395.5	77300	316.6	79800	237.6
72400	354.1	74900	275.2	77400	359.0	79900	320.7
72500	437.2	75000	358.3	77500	432.4	80000	363.1

Table A.17. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
80100	283.5	82600	367.3	85100	299.0	87600	229.8
80200	346.3	82700	287.7	85200	361.8	87700	312.9
80300	388.7	82800	330.1	85300	454.6	87800	355.3
80400	288.7	82900	372.5	85400	354.6	87900	266.0
80500	371.8	83000	292.8	85500	397.0	88000	358.8
80600	414.2	83100	355.6	85600	317.3	88100	238.4
80700	314.2	83200	275.9	85700	359.8	88200	341.9
80800	367.3	83300	338.7	85800	442.9	88300	343.6
80900	297.3	83400	360.8	85900	342.9	88400	284.3
81000	360.1	83500	301.5	86000	405.6	88500	306.3
81100	432.5	83600	354.5	86100	255.3	88600	389.5
81200	313.1	83700	264.2	86200	348.0	88700	289.5
81300	365.2	83800	306.6	86300	390.5	88800	331.9
81400	448.3	83900	389.7	86400	331.1	88900	435.3
81500	348.3	84000	269.4	86500	393.9	89000	294.6
81600	390.8	84100	352.5	86600	263.9	89100	347.7
81700	311.1	84200	394.9	86700	336.3	89200	277.7
81800	353.5	84300	285.2	86800	399.1	89300	320.1
81900	426.9	84400	378.0	86900	319.4	89400	423.6
82000	316.3	84500	278.0	87000	341.5	89500	282.9
82100	379.0	84600	340.8	87100	241.5	89600	356.3
82200	279.0	84700	383.2	87200	345.0	89700	428.8
82300	382.5	84800	283.2	87300	367.0	89800	308.4
82400	424.9	84900	366.3	87400	307.7	89900	391.5
82500	324.9	85000	429.1	87500	350.1	90000	454.3

Table A.18. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
90100	354.3	92600	265.7	95100	379.5	97600	341.3
90200	417.1	92700	358.5	95200	299.9	97700	383.7
90300	317.1	92800	258.5	95300	332.6	97800	304.0
90400	379.8	92900	300.9	95400	384.7	97900	346.4
90500	442.6	93000	384.0	95500	305.0	98000	388.9
90600	342.6	93100	284.0	95600	367.8	98100	329.6
90700	425.7	93200	326.4	95700	288.1	98200	372.0
90800	305.3	93300	409.5	95800	330.6	98300	455.1
90900	347.8	93400	299.8	95900	423.4	98400	355.1
91000	410.5	93500	372.3	96000	293.3	98500	417.8
91100	310.5	93600	435.0	96100	356.1	98600	277.1
91200	393.6	93700	335.0	96200	256.1	98700	360.3
91300	293.6	93800	377.4	96300	318.8	98800	443.4
91400	356.4	93900	297.8	96400	402.0	98900	343.4
91500	398.8	94000	360.5	96500	292.3	99000	365.4
91600	298.8	94100	423.3	96600	344.4	99100	306.1
91700	381.9	94200	323.3	96700	244.4	99200	348.5
91800	281.9	94300	365.7	96800	307.1	99300	411.3
91900	324.3	94400	428.5	96900	369.9	99400	331.6
92000	407.4	94500	328.5	97000	269.9	99500	374.1
92100	287.1	94600	411.6	97100	332.6	99600	274.1
92200	370.2	94700	301.9	97200	405.1	99700	336.8
92300	270.2	94800	354.0	97300	295.4	99800	419.9
92400	332.9	94900	396.4	97400	337.8	99900	279.2
92500	395.7	95000	327.4	97500	278.5	100000	373.0

Table A.19. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
100100	262.3	102600	366.5	105100	287.6	107600	412.1
100200	315.4	102700	399.2	105200	350.3	107700	312.1
100300	368.5	102800	329.3	105300	250.3	107800	364.2
100400	308.2	102900	371.7	105400	333.4	107900	407.6
100500	330.3	103000	454.8	105500	396.2	108000	317.3
100600	271.0	103100	354.8	105600	296.2	108100	400.4
100700	313.4	103200	437.9	105700	338.6	108200	280.0
100800	396.5	103300	297.2	105800	238.6	108300	363.1
100900	276.1	103400	380.3	105900	321.7	108400	425.9
101000	338.9	103500	443.1	106000	364.1	108500	325.9
101100	259.2	103600	343.1	106100	284.5	108600	368.3
101200	301.7	103700	405.8	106200	347.2	108700	268.3
101300	364.4	103800	305.8	106300	410.0	108800	331.1
101400	244.1	103900	348.2	106400	310.0	108900	414.2
101500	347.5	104000	421.7	106500	372.8	109000	314.2
101600	389.9	104100	331.3	106600	435.5	109100	376.9
101700	289.9	104200	394.1	106700	315.2	109200	276.9
101800	352.7	104300	294.1	106800	398.3	109300	319.3
101900	405.8	104400	356.9	106900	298.3	109400	382.1
102000	335.8	104500	419.6	107000	320.3	109500	261.8
102100	378.2	104600	299.3	107100	403.5	109600	344.9
102200	257.9	104700	382.4	107200	303.5	109700	265.2
102300	341.0	104800	282.4	107300	386.6	109800	307.6
102400	403.7	104900	324.8	107400	449.3	109900	370.4
102500	324.1	105000	387.6	107500	349.3	110000	270.4

Table A.20. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
110100	353.5	112600	274.5	115100	378.7	117600	320.1
110200	416.2	112700	337.3	115200	278.7	117700	382.9
110300	295.9	112800	420.4	115300	341.5	117800	282.9
110400	379.0	112900	320.4	115400	404.2	117900	345.6
110500	269.3	113000	342.5	115500	304.2	118000	408.4
110600	321.4	113100	283.2	115600	357.3	118100	308.4
110700	404.5	113200	325.6	115700	429.8	118200	350.8
110800	284.2	113300	378.7	115800	329.8	118300	271.2
110900	367.3	113400	308.7	115900	372.2	118400	333.9
111000	430.0	113500	371.4	116000	455.3	118500	366.6
111100	289.4	113600	260.8	116100	334.9	118600	276.3
111200	362.8	113700	313.9	116200	418.0	118700	359.4
111300	455.6	113800	356.3	116300	318.0	118800	259.4
111400	355.6	113900	297.0	116400	380.8	118900	322.2
111500	398.0	114000	319.0	116500	443.6	119000	364.6
111600	288.3	114100	239.4	116600	343.6	119100	264.6
111700	360.7	114200	302.1	116700	406.3	119200	317.7
111800	403.2	114300	385.3	116800	296.6	119300	410.5
111900	323.5	114400	264.9	116900	369.1	119400	290.1
112000	386.3	114500	327.7	117000	431.8	119500	373.2
112100	265.9	114600	370.1	117100	331.8	119600	436.0
112200	349.0	114700	290.4	117200	374.3	119700	326.3
112300	391.4	114800	332.8	117300	294.6	119800	378.4
112400	332.1	114900	395.6	117400	337.0	119900	298.8
112500	394.9	115000	336.3	117500	420.1	120000	351.8

Table A.21. 100μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
120100	424.3	122600	325.0	125100	307.1
120200	324.3	122700	225.0	125200	339.8
120300	366.7	122800	328.5	125300	391.9
120400	449.8	122900	370.9	125400	332.6
120500	349.8	123000	291.2	125500	375.0
120600	412.6	123100	354.0	125600	254.7
120700	292.2	123200	416.7	125700	378.5
120800	355.0	123300	296.4	125800	380.2
120900	397.4	123400	359.1	125900	320.9
121000	317.7	123500	279.5	126000	363.3
121100	380.5	123600	321.9	126100	243.0
121200	280.5	123700	405.0	126200	326.1
121300	343.3	123800	305.0	126300	388.8
121400	426.4	123900	327.1	126400	268.5
121500	326.4	124000	410.2	126500	371.9
121600	389.1	124100	310.2	126600	231.2
121700	268.8	124200	372.9	126700	314.3
121800	351.9	124300	456.1	126800	377.1
121900	414.6	124400	335.7	126900	247.1
122000	314.6	124500	378.1	127000	319.5
122100	377.4	124600	318.8	127100	412.3
122200	257.1	124700	361.2	127200	302.6
122300	340.2	124800	424.0		
122400	402.9	124900	303.6		
122500	282.6	125000	386.7		

Table A.22. 1000μs data

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
1000	369.6	26000	387.1	51000	452.8	76000	294.1
2000	340.3	27000	353.6	52000	429.4	77000	453.8
3000	313.0	28000	330.2	53000	406.0	78000	430.4
4000	289.0	29000	292.2	54000	362.2	79000	386.6
5000	277.0	30000	302.7	55000	349.4	80000	373.8
6000	237.6	31000	251.8	56000	294.9	81000	339.7
7000	393.4	32000	208.3	57000	271.5	82000	316.3
8000	346.3	33000	364.1	58000	288.8	83000	313.2
9000	340.6	34000	361.0	59000	387.4	84000	249.1
10000	318.7	35000	296.9	60000	404.7	85000	408.7
11000	287.6	36000	284.1	61000	381.2	86000	426.0
12000	239.8	37000	292.7	62000	317.1	87000	341.5
13000	415.3	38000	309.9	63000	334.3	88000	318.1
14000	359.9	39000	428.9	64000	433.0	89000	315.0
15000	336.5	40000	385.1	65000	429.9	90000	433.9
16000	319.2	41000	361.7	66000	406.4	91000	410.5
17000	302.2	42000	338.2	67000	383.0	92000	387.1
18000	262.3	43000	314.8	68000	359.6	93000	384.0
19000	255.3	44000	261.3	69000	336.1	94000	340.2
20000	401.4	45000	267.9	70000	312.7	95000	316.8
21000	347.9	46000	427.6	71000	289.3	96000	293.3
22000	344.9	47000	363.5	72000	387.9	97000	290.2
23000	324.0	48000	340.0	73000	405.1	98000	429.6
24000	284.4	49000	316.6	74000	361.4	99000	385.8
25000	260.0	50000	333.9	75000	358.3	100000	342.0

Table A.23. 1000μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
101000	359.2	126000	343.0	151000	387.7	176000	402.5
102000	295.1	127000	339.9	152000	343.9	177000	409.0
103000	454.8	128000	336.8	153000	340.8	178000	344.9
104000	431.3	129000	455.7	154000	307.7	179000	341.8
105000	387.6	130000	432.3	155000	426.7	180000	338.7
106000	343.8	131000	408.9	156000	412.9	181000	448.0
107000	310.7	132000	385.4	157000	389.5	182000	413.9
108000	337.6	133000	352.3	158000	386.4	183000	410.8
109000	273.5	134000	319.2	159000	322.3	184000	337.0
110000	290.7	135000	274.4	160000	339.5	185000	323.3
111000	409.7	136000	251.0	161000	316.1	186000	320.2
112000	406.6	137000	390.3	162000	292.7	187000	317.1
113000	362.8	138000	407.6	163000	452.3	188000	252.9
114000	359.7	139000	354.1	164000	367.9	189000	433.0
115000	316.0	140000	340.3	165000	385.1	190000	368.8
116000	455.3	141000	337.3	166000	321.0	191000	386.1
117000	411.5	142000	435.9	167000	317.9	192000	362.7
118000	388.0	143000	432.8	168000	457.2	193000	298.5
119000	344.3	144000	409.4	169000	433.8	194000	275.1
120000	320.8	145000	385.9	170000	410.3	195000	434.8
121000	338.1	146000	362.5	171000	386.9	196000	411.3
122000	314.6	147000	339.0	172000	363.5	197000	387.9
123000	281.5	148000	315.6	173000	340.0	198000	364.4
124000	389.8	149000	251.5	174000	275.9	199000	300.3
125000	407.1	150000	411.1	175000	293.1	200000	276.9

Table A.24. 1000μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
201000	294.1	226000	298.2	251000	302.3	276000	367.4
202000	433.5	227000	254.4	252000	319.5	277000	303.3
203000	389.7	228000	393.7	253000	296.1	278000	300.1
204000	386.6	229000	411.0	254000	415.1	279000	256.4
205000	342.8	230000	367.2	255000	412.0	280000	436.4
206000	319.3	231000	323.4	256000	368.2	281000	372.3
207000	295.9	232000	340.7	257000	324.4	282000	369.2
208000	435.2	233000	296.9	258000	301.0	283000	325.4
209000	371.1	234000	436.2	259000	288.2	284000	322.3
210000	378.7	235000	372.1	260000	396.5	285000	278.5
211000	364.9	236000	389.3	261000	413.8	286000	438.2
212000	341.5	237000	365.9	262000	380.6	287000	394.4
213000	318.0	238000	342.5	263000	326.2	288000	380.6
214000	253.9	239000	319.0	264000	343.4	289000	367.8
215000	433.9	240000	275.3	265000	279.3	290000	303.8
216000	369.8	241000	414.6	266000	276.2	291000	300.6
217000	366.7	242000	411.5	267000	415.6	292000	256.9
218000	343.3	243000	367.7	268000	392.1	293000	436.8
219000	319.8	244000	344.3	269000	368.7	294000	372.8
220000	276.1	245000	300.5	270000	324.9	295000	369.7
221000	435.7	246000	297.4	271000	321.8	296000	346.2
222000	371.6	247000	436.7	272000	298.4	297000	322.8
223000	388.8	248000	372.6	273000	437.7	298000	279.0
224000	365.4	249000	389.8	274000	414.3	299000	418.3
225000	321.6	250000	366.4	275000	390.8	300000	394.9

Table A.25. 1000μs data (cont)

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
301000	371.4	326000	395.8	351000	440.6
302000	368.3	327000	392.8	352000	417.2
303000	304.2	328000	369.3	353000	383.1
304000	321.5	329000	325.5	354000	349.9
305000	257.3	330000	281.8	355000	316.8
306000	437.3	331000	299.0	356000	323.4
307000	393.6	332000	255.2	357000	270.0
308000	360.5	333000	374.2	358000	256.2
309000	326.3	334000	350.8	359000	375.2
310000	323.3	335000	347.7	360000	372.1
311000	299.8	336000	324.3	361000	348.7
312000	418.8	337000	280.4	362000	325.2
313000	415.7	338000	399.4	363000	271.8
314000	392.3	339000	416.7	364000	400.4
315000	328.2	340000	393.3		
316000	345.4	341000	329.1		
317000	281.3	342000	346.4		
318000	298.5	343000	272.6		
319000	254.7	344000	269.5		
320000	373.7	345000	276.1		
321000	350.3	346000	395.0		
322000	336.5	347000	371.6		
323000	303.4	348000	348.2		
324000	280.0	349000	304.4		
325000	419.3	350000	280.9		

Table A.26. 10,000μs data

Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)	Delay Request (μs)	Additional Delay (μs)
10000	311.5	260000	437.2	510000	355.9	760000	294.9
20000	384.9	270000	345.3	520000	426.7	770000	345.3
30000	268.8	280000	416.0	530000	334.7	780000	416.1
40000	405.5	290000	324.1	540000	242.9	790000	324.2
50000	313.5	300000	415.2	550000	313.6	800000	415.4
60000	384.3	310000	323.3	560000	404.7	810000	323.3
70000	303.0	320000	394.1	570000	312.8	820000	394.1
80000	383.5	330000	302.1	580000	383.6	830000	322.5
90000	454.3	340000	393.3	590000	311.9	840000	393.3
100000	362.3	350000	280.9	600000	382.9	850000	301.4
110000	290.7	360000	351.8	610000	281.1	860000	392.6
120000	361.5	370000	259.8	620000	361.6	870000	300.6
130000	432.3	380000	371.3	630000	269.6	880000	371.3
140000	340.3	390000	442.1	640000	340.4	890000	442.1
150000	431.5	400000	329.8	650000	431.6	900000	329.8
160000	319.2	410000	237.8	660000	339.7	910000	278.7
170000	369.6	420000	349.3	670000	227.4	920000	339.7
180000	338.7	430000	379.4	680000	318.4	930000	420.2
190000	368.8	440000	328.2	690000	389.2	940000	307.9
200000	297.2	450000	378.6	700000	317.6	950000	389.3
210000	347.7	460000	286.7	710000	388.4	960000	286.7
220000	296.4	470000	357.4	720000	296.4	970000	398.2
230000	387.5	480000	296.5	730000	367.4	980000	306.3
240000	265.6	490000	356.6	740000	295.6	990000	397.5
250000	325.7	500000	285.1	750000	366.4		

Appendix B. *System Clock Update Analysis*

B.1 Overview

The XD Ada MC68020 run-time system clock is updated every $162.5\mu s$ and generates an interrupt that the runtime system must handle every $256 \cdot 162.5\mu s = 41,600\mu s$. Since the clock interrupt handler is short and there are only two execution paths, it readily lends itself to manual analysis. The interrupt handler is reproduced in Section B.3 (used by permission (13)). To insure the worst case execution time, the analysis assumed the worst case execution path. The number of clock cycles to execute a given instruction is contained within square braces (e.g. [ ]). Since the MC68020 is a pipelined architecture and there is no method of predicting the state of the pipeline at an arbitrary point in time, worst case execution times are assumed. Also note that the MVME133A-20 single board computer inserts 1 wait state for every memory reference (14:31). This wait state is accounted for in the execution time indicated.

B.2 Interrupt Response Time

The response time for an M-Stack interrupt is 48 clock cycles (15:10-40). In addition, there are 12 memory references associated with the interrupt response cycle which results in 12 wait states. Therefore, the worst case interrupt response time is 60 clock cycles once the interrupt has been recognized. Actually, it may take significantly longer for the clock interrupt to be recognized due to hardware interrupt priorities and the fact that pending interrupts are only recognized after the execution of the current instruction. The MOVEM instruction, however, is an exception to this rule. Pending interrupts are not recognized after the execution of a MOVEM instruction, but rather recognition is delayed until the instruction following MOVEM.

This interrupt recognition latency has not been included in this analysis since, in the context of this research, it does not induce any blocking with respect to RMA. Only the actual interrupt

response time and interrupt handler block a user task. The time to recognize that an interrupt occurred, in fact, does not penalize the user task whatsoever.

```
* ------------------------------------------------------------------------------ *
* --      X     X    DDDDD                AAA      DDDDD       AAA         -- *
* --       X   X     D    D              A   A     D    D      A   A        -- *
* --        X X      D     D            A     A    D     D    A     A       -- *
* --         X       D     D            AAAAAAA    D     D    AAAAAAA       -- *
* --        X X      D     D            A     A    D     D    A     A       -- *
* --       X   X     D    D             A     A    D    D     A     A       -- *
* --      X     X    DDDDD              A     A    DDDDD      A     A       -- *
* ------------------------------------------------------------------------------ *
*
* COPYRIGHT (c) 1988,1991 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
* ALL RIGHTS RESERVED.
* COPYRIGHT (c) 1988,1991 BY
* SD-SCICON PLC, FLEET, HAMPSHIRE, ENGLAND.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND  COPIED
* ONLY  IN  ACCORDANCE  WITH  THE  TERMS  OF  SUCH  LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE  WITHOUT  NOTICE
* AND  SHOULD  NOT  BE  CONSTRUED  AS  A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION OR SD-SCICON UK LTD.
*
* ------------------------------------------------------------------------------ *

                        TITLE       "CLOCK"
                        MODULE      "CLOCK"
                        IDENT       "V1.2A-33"

*     This module provides the run-time system support for package
*     CALENDAR, and provides time-out support for the TASKING system.
*     It implements the clock interrupt handlers, together with routines
*     to initialise the timer(s) and return the current time. It also
*     contains routines which are called from the Target Kernel to
*     stop and restart the clock.

* ------------------------------------------------------------------------------
*--    procedure COMMON_HANDLER is
*--    begin

*--       -- This routine is the interrupt handler for both the chiming
*--       -- clock and the alarm clock if both exist, both use the same
*--       -- interrupt vector, and tasking is present.

*--       if GL$_DELAY_TIMER >= 0 then
```

```
*--          CHIME_HANDLER;
*--       else
*--          ALARM_HANDLER;
*--       end if;

*--    end COMMON_HANDLER;

COMMON_HANDLER:
                        TST.L       GL$_DELAY_TIMER.L  [13]
                        BGE.W       CHIME_HANDLER      [11, branch taken]
                        BRA.W       ALARM_HANDLER

* ------------------------------------------------------------------------------

*--    procedure CLR_CHIME;
*--    pragma INLINE (CLR_CHIME);

*--    -- This macro need only be provided if a chiming clock is used.
*--    -- MACRO CLR_CHIME is called as the first action within the
*--    -- chiming timer's interrupt handler. It should clear the chiming
*--    -- timer's interrupt. In addition, if both a chiming clock and an
*--    -- alarm clock are being used, it is recommended that this macro
*--    -- also makes sure that the alarm timer is stopped. This macro
*--    -- must not modify any registers.

CLR_CHIME:              MACRO
                        ANDI.B      #$F8,CL$_TCDCR [17]
                        ENDMAC

* ------------------------------------------------------------------------------

*--    procedure SET_ALARM (TICKS : in INTEGER);
*--    pragma INLINE (SET_ALARM);

*--    -- This macro need only be provided if an alarm clock is used.
*--    -- It should set the alarm clock to interrupt after TICKS
*--    -- SYSTEM.TICKs, where TICKS is always contained in register
*--    -- D0.L. TICKS will always be in the range 1..MAX_ALARM. This
*--    -- macro must not modify any registers other than D0.

SET_ALARM:              MACRO
                        ANDI.B      #$F8,CL$_TCDCR  [17]
                        MOVE.B      D0,CL$_TDDR     [9]
                        ORI.B       #$07,CL$_TCDCR  [17]
                        ENDMAC

* ------------------------------------------------------------------------------
*--    procedure CHIME_HANDLER is
*--    begin

*--       -- This routine is the chiming clock interrupt handler if both a
```

```
*--       -- chiming clock and an alarm clock exist, tasking is present,
*--       -- and MAX_ALARM is greater than or equal to CHIME_PERIOD.

*--       CLR_CHIME;
*--       GL$_CLOCK := GL$_CLOCK + CHIME_PERIOD;
*--       if GL$_DELAY_TIMER > 0 then
*--          if GL$_DELAY_TIMER < CHIME_PERIOD then
*--             SET_ALARM (GL$_DELAY_TIMER);
*--          end if;
*--          GL$_DELAY_TIMER := GL$_DELAY_TIMER - CHIME_PERIOD;
*--       else
*--          GL$_REQUESTS (T$_RQ_ALARM) := TRUE;
*--       end if;
*--    end CHIME_HANDLER;

CHIME_HANDLER:
                        CLR_CHIME                                [17]
                        ADDI.L      #CHIME_PERIOD,GL$_LS_CLOCK.L [23]
                        BCC.B       CH_NO_CARRY1                 [6, not taken]
                        ADDQ.L      #1,GL$_MS_CLOCK.L            [17]
CH_NO_CARRY1:           MOVE.L      D0,-(A7)                     [8]
CH_IF1:                 MOVE.L      GL$_DELAY_TIMER.L,D0         [13]
                        BLE.W       CH_ELSE1                     [8, not taken]
CH_THEN1:
CH_IF2:                 CMPI.L      #CHIME_PERIOD,D0             [10]
                        BGE.W       CH_ENDIF2                    [8, not taken]
CH_THEN2:               SET_ALARM                                [43]
CH_ENDIF2:
                        SUBI.L      #CHIME_PERIOD,
                                                GL$_DELAY_TIMER.L [18]
                        BRA.B       CH_ENDIF1                    [11, taken]
CH_ELSE1:               BSET.B      #T$_RQ_ALARM,GL$_REQUESTS.L
CH_ENDIF1:
                        MOVE.L      (A7)+,D0                     [9]
                        MOVE.L      GL$_INT_RETURN.L,-(A7)       [18]
                        RTS                                      [15]
```

B.4 Clock Update Analysis

The number of clock cycles required to update the system clock and return is the summation of the interrupt response time (60 clock cycles) and the interrupt handler (248 clock cycles). Each clock cycle takes $\frac{1}{20MHz} = 0.05\mu s$. Therefore, the time required to update the system clock is:

$$(248 + 60) \times 0.05\mu s = 15.4\mu s$$

Since the clock is updated every $41600\mu s$, the worst case CPU utilization for clock updates is $\frac{15.4}{41600} = 0.0370\%$.

Appendix C. *Hartstone/RATESIM Validation Data*

This appendix contains the raw validation data gathered during the testing of the RATESIM model. Each section contains the summary Hartstone benchmark results and three RATESIM model runs. The first RATESIM run is of the last point in which RATESIM determined that the user task set was schedulable, the second run in the first point at which the user task set failed, and the last run is of the user task set using the same task parameters at which the task set failed on the target hardware while running the Hartstone benchmark.

C.1 Task Set A - Harmonic

C.1.1 Hartstone Results - Experiment 1

HARTSTONE BENCHMARK SUMMARY RESULTS

Baseline test:

==========

Experiment: EXPERIMENT_1 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.78

Test 1 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	4.00	16	64.00	4.79 %
3	8.00	8	64.00	4.79 %
4	16.00	4	64.00	4.79 %
5	32.00	2	64.00	4.79 %
			320.00	23.94 %

Experiment step size: 2.39 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_1 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.78

Test 24 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	4.00	16	64.00	4.79 %
3	8.00	8	64.00	4.79 %
4	16.00	4	64.00	4.79 %
5	400.00	2	800.00	59.85 %
			1056.00	79.00 %

Experiment step size: 2.39 %

Test 24 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	2.500	4001	0	0	0.000

```
================================================================================

Test when deadlines first missed/skipped:

================================================================================

Experiment:    EXPERIMENT_1 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.78

Test 25 characteristics:

   Task    Frequency    Kilo-Whets    Kilo-Whets    Requested Workload
    No.    (Hertz)      per period    per second       Utilization
     1        2.00          32          64.00            4.79 %
     2        4.00          16          64.00            4.79 %
     3        8.00           8          64.00            4.79 %
     4       16.00           4          64.00            4.79 %
     5      416.00           2         832.00           62.24 %
                                       -------          -------
                                      1088.00           81.39 %

Experiment step size:   2.39 %

--------------------------------------------------------------------------------

Test 25 results:

Test duration (seconds):  10.0

   Task     Period        Met         Missed       Skipped       Average
    No.     in msecs    Deadlines     Deadlines    Deadlines     Late (msec)
     1      500.000         0            10           10          472.650
     2      250.000        40             0            0            0.000
     3      125.000        80             0            0            0.000
     4       62.500       160             0            0            0.000
     5        2.404      4158             1            1            0.061

================================================================================

Final test performed:

================================================================================

Experiment:    EXPERIMENT_1 HARMONIC
```

Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.78

Test 26 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	4.00	16	64.00	4.79 %
3	8.00	8	64.00	4.79 %
4	16.00	4	64.00	4.79 %
5	432.00	2	864.00	64.63 %
			1120.00	83.78 %

Experiment step size: 2.39 %

Test 26 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	0	7	13	791.190
2	250.000	4	18	18	79.376
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	2.315	4314	3	3	0.163

==========

==========

Benchmark : Hartstone Benchmark, Version 1.0
Compiler : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

Test 24 of Experiment 1 Harmonic

Raw (non-tasking) benchmark speed in KWIPS: 1336.78

Full task set:

```
   Total       Deadlines        Task Set        Total
   Tasks       Per Second      Utilization      KWIPS
     5          430.00          79.00 %       1056.00

  Highest-frequency task:

   Period      Deadlines         Task           Task
   (msec)      Per Second      Utilization      KWIPS
    2.500       400.00          59.85 %        800.00

  Experiment step size:   2.39 %

================================================================================

              END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.1.2 RATESIM Results - Experiment 1

C.1.2.1 Successful Scheduling

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp1_pass
                      Execution
Task name              Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------             --------- ------------------------  ------------
Task 5                   1496      2458    /   406.83         2458
   Rendezvous        : none


Task 4                   2992     62500    /    16.00        62500
   Rendezvous        : none


Task 3                   5984    125000    /     8.00       125000
   Rendezvous        : none
```

```
Task 2                    11969     250000    /     4.00        250000
   Rendezvous         : none

Task 1                    23938     500000    /     2.00        500000
   Rendezvous         : none


================================================================================


                        --------------------------------
                        |Rate Monotonic Scheduler Model|
                        --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                6086302
Deadlines Met :                                   4068
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           268
Worst case blocking time in a single period (us):
                                                   922
Cumulative early deadlines (us):            2209876.00
Context Switches :                                4336
Delay Expirations :                               4068

=====================================================


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                 478720
```

```
Deadlines Met :                                    160
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           977
Worst case blocking time in a single period (us):
                                                  3801
Cumulative early deadlines (us):            7438185.00
Context Switches :                                1137
Delay Expirations :                                159

============================================================


============================================================

Task Statistics for task : Task 3
------------------------------------------------------------
Cumulative Execution_Time (us):                 478720
Deadlines Met :                                    80
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1027
Worst case blocking time in a single period (us):
                                                  9592
Cumulative early deadlines (us):            6266005.00
Context Switches :                                1107
Delay Expirations :                                79

============================================================


============================================================

Task Statistics for task : Task 2
------------------------------------------------------------
Cumulative Execution_Time (us):                 478760
Deadlines Met :                                    40
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1034
Worst case blocking time in a single period (us):
                                                 34365
Cumulative early deadlines (us):            4965246.00
Context Switches :                                1074
Delay Expirations :                                39

============================================================


============================================================
```

```
Task Statistics for task : Task 1
--------------------------------------------------
Cumulative Execution_Time (us):                478760
Deadlines Met :                                    20
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :         1076
Worst case blocking time in a single period (us):
                                                99178
Cumulative early deadlines (us):              87792.00
Context Switches :                                1096
Delay Expirations :                                 19

==================================================


Simulation Time (us):                       10000058
User Cumulative Task Execution Time (us):    8001262
User Deadlines Met :                            4368
User Deadlines Missed :                            0
Context Switches :                              8750
Delay Expirations :                             4364
Rendezvous executed :                              0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):   405446
System Task Execution Time (us):            1977526
Idle Time (us):                               21270
Percentage User Task Execution Time :       80.012156
Percentage System Task Execution :          19.775145
Percentage Idle Time :                       0.212699

================================================================================
```

C.1.2.2 Scheduling Failure- Experiment 1

```
                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp1_fail

                        Execution
```

```
Task name              Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------              --------- ------------------------  ------------
Task 5                    1496      2457    /   407.00         2457
   Rendezvous        : none

Task 4                    2992     62500    /    16.00        62500
   Rendezvous        : none

Task 3                    5984    125000    /     8.00       125000
   Rendezvous        : none

Task 2                   11969    250000    /     4.00       250000
   Rendezvous        : none

Task 1                   23938    500000    /     2.00       500000
   Rendezvous        : none

================================================================================

                      ---------------------------------
                      |Rate Monotonic Scheduler Model|
                      ---------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


======================================================

Task Statistics for task : Task 5
------------------------------------------------------
Cumulative Execution_Time (us):                6088720
Deadlines Met :                                   4068
Deadlines Missed :                                   2
First deadline missed at :                     4501224
Execution completed at :                       4501307
Cumulative late deadlines (us):                 154.00
Preemptions suffered due to higher
```

```
   priority user tasks or system tasks :            274
Worst case blocking time in a single period (us):
                                                   1044
Cumulative early deadlines (us):            2215230.00
Context Switches :                                 4342
Delay Expirations :                                4067

============================================================


============================================================

Task Statistics for task : Task 4
------------------------------------------------------------
Cumulative Execution_Time (us):                  478720
Deadlines Met :                                     160
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            967
Worst case blocking time in a single period (us):
                                                   3833
Cumulative early deadlines (us):            7456530.00
Context Switches :                                 1127
Delay Expirations :                                 159

============================================================


============================================================

Task Statistics for task : Task 3
------------------------------------------------------------
Cumulative Execution_Time (us):                  478720
Deadlines Met :                                      80
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :           1037
Worst case blocking time in a single period (us):
                                                   9554
Cumulative early deadlines (us):            6253632.00
Context Switches :                                 1117
Delay Expirations :                                  79

============================================================


============================================================

Task Statistics for task : Task 2
------------------------------------------------------------
Cumulative Execution_Time (us):                  478760
```

```
Deadlines Met :                                    40
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1025
Worst case blocking time in a single period (us):
                                                24664
Cumulative early deadlines (us):             5053884.00
Context Switches :                                1065
Delay Expirations :                                 39

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 478760
Deadlines Met :                                     20
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1091
Worst case blocking time in a single period (us):
                                                98981
Cumulative early deadlines (us):              66821.00
Context Switches :                                1111
Delay Expirations :                                 19

=======================================================


Simulation Time (us):                         10000001
User Cumulative Task Execution Time (us):      8003680
User Deadlines Met :                              4368
User Deadlines Missed :                              2
Context Switches :                                8760
Delay Expirations :                               4363
Rendezvous executed :                                0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):     396314
System Task Execution Time (us):              1978702
Idle Time (us):                                 17611
Percentage User Task Execution Time :         80.036792
Percentage System Task Execution :            19.787018
Percentage Idle Time :                         0.176110

=============================================================================
```

C.1.2.3 Scheduling Failure - Experiment 1(Hartstone Benchmark Task Parameters)

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp1_fail.hart

                     Execution
Task name             Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------            --------- ------------------------  ------------
Task 5                   1496      2404    /   415.97         2404
   Rendezvous        : none


Task 4                   2992     62500    /    16.00        62500
   Rendezvous        : none


Task 3                   5984    125000    /     8.00       125000
   Rendezvous        : none


Task 2                  11969    250000    /     4.00       250000
   Rendezvous        : none


Task 1                  23938    500000    /     2.00       500000
   Rendezvous        : none


================================================================================


                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
```

```
Enter length of simulation in microseconds:10_000_000
Print the Event History (y or n) : n

==========================================================

Task Statistics for task : Task 5
----------------------------------------------------------
Cumulative Execution_Time (us):                  6223360
Deadlines Met :                                     4159
Deadlines Missed :                                     1
First deadline missed at :                       8688056
Execution completed at :                         8688063
Cumulative late deadlines (us):                     7.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             192
Worst case blocking time in a single period (us):
                                                      804
Cumulative early deadlines (us):              2010855.00
Context Switches :                                  4351
Delay Expirations :                                 4158

==========================================================

==========================================================

Task Statistics for task : Task 4
----------------------------------------------------------
Cumulative Execution_Time (us):                   478896
Deadlines Met :                                      160
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :            1065
Worst case blocking time in a single period (us):
                                                     4196
Cumulative early deadlines (us):              7297544.00
Context Switches :                                   1225
Delay Expirations :                                  160

==========================================================

==========================================================

Task Statistics for task : Task 3
----------------------------------------------------------
Cumulative Execution_Time (us):                   478720
Deadlines Met :                                       80
Deadlines Missed :                                     0
Preemptions suffered due to higher
```

```
   priority user tasks or system tasks :          1139
Worst case blocking time in a single period (us):
                                                 10372
Cumulative early deadlines (us):            5972978.00
Context Switches :                               1219
Delay Expirations :                                80

============================================================


============================================================

Task Statistics for task : Task 2
------------------------------------------------------------
Cumulative Execution_Time (us):                478760
Deadlines Met :                                    40
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1235
Worst case blocking time in a single period (us):
                                                 42487
Cumulative early deadlines (us):            2317635.00
Context Switches :                               1275
Delay Expirations :                                40

============================================================


============================================================

Task Statistics for task : Task 1
------------------------------------------------------------
Cumulative Execution_Time (us):                330520
Deadlines Met :                                     0
Deadlines Missed :                                 13
First deadline missed at :                     500000
Execution completed at :                       733014
Cumulative late deadlines (us):            19673080.00
Preemptions suffered due to higher
   priority user tasks or system tasks :           863
Worst case blocking time in a single period (us):
                                                153315
Cumulative early deadlines (us):                 0.00
Context Switches :                                863
Delay Expirations :                                 0

============================================================


Simulation Time (us):                       10000755
User Cumulative Task Execution Time (us):    7990256
```

```
User Deadlines Met :                                4439
User Deadlines Missed :                               14
Context Switches :                                  8919
Delay Expirations :                                 4438
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):      474816
System Task Execution Time (us):                 2010443
Idle Time (us):                                        0
Percentage User Task Execution Time :           79.896528
Percentage System Task Execution :              20.102912
Percentage Idle Time :                           0.000000

================================================================================
```

C.1.3 Hartstone Results - Experiment 2

```
                    HARTSTONE BENCHMARK SUMMARY RESULTS

Baseline test:

================================================================================

Experiment:    EXPERIMENT_2 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 1 characteristics:

   Task    Frequency    Kilo-Whets    Kilo-Whets    Requested Workload
    No.     (Hertz)     per period    per second       Utilization
     1        2.00          32          64.00            4.79 %
     2        4.00          16          64.00            4.79 %
     3        8.00           8          64.00            4.79 %
     4       16.00           4          64.00            4.79 %
     5       32.00           2          64.00            4.79 %
                                       -------          --------
                                       320.00           23.94 %

Experiment step size:   2.39 %

--------------------------------------------------------------------------------

Test 1 results:

Test duration (seconds):  10.0
```

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_2 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 29 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	7.60	32	243.20	18.19 %
2	15.20	16	243.20	18.19 %
3	30.40	8	243.20	18.19 %
4	60.80	4	243.20	18.19 %
5	121.60	2	243.20	18.19 %
			1216.00	90.96 %

Experiment step size: 2.39 %

Test 29 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	131.579	76	0	0	0.000
2	65.789	152	0	0	0.000
3	32.895	304	0	0	0.000
4	16.447	608	0	0	0.000
5	8.224	1216	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_2 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 30 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	7.80	32	249.60	18.67 %
2	15.60	16	249.60	18.67 %
3	31.20	8	249.60	18.67 %
4	62.40	4	249.60	18.67 %
5	124.80	2	249.60	18.67 %
			1248.00	93.36 %

Experiment step size: 2.39 %

Test 30 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	128.205	1	39	39	49.521
2	64.103	157	0	0	0.000
3	32.051	313	0	0	0.000
4	16.026	625	0	0	0.000
5	8.013	1249	0	0	0.000

==========

Final test performed:
See preceding summary of test 30

==========

```
Benchmark : Hartstone Benchmark, Version 1.0
Compiler  : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target    : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

   Test 29 of Experiment 2 Harmonic

   Raw (non-tasking) benchmark speed in KWIPS: 1336.80

   Full task set:

      Total       Deadlines        Task Set        Total
      Tasks       Per Second      Utilization      KWIPS
        5           235.60          90.96 %      1216.00

   Highest-frequency task:

      Period      Deadlines          Task          Task
      (msec)      Per Second      Utilization      KWIPS
       8.224        121.60          18.19 %       243.20

   Experiment step size:   2.39 %

==========================================================================

              END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.1.4 RATESIM Results - Experiment 2

C.1.4.1 Successful Scheduling

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit
```

```
Enter choice: g
================================================================================
Enter file name to get the task set from: exp2_pass
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------- ------------
Task 5                      1496      8123    /   123.11         8123
   Rendezvous         : none

Task 4                      2992     16246    /    61.55        16246
   Rendezvous         : none

Task 3                      5984     32492    /    30.78        32492
   Rendezvous         : none

Task 2                     11969     64984    /    15.39        64984
   Rendezvous         : none

Task 1                     23938    129968    /     7.69       129968
   Rendezvous         : none

================================================================================

                      -------------------------------
                      |Rate Monotonic Scheduler Model|
                      -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


==========================================================

Task Statistics for task : Task 5
----------------------------------------------------------
Cumulative Execution_Time (us):                  1843072
Deadlines Met :                                     1232
Deadlines Missed :                                     0
```

```
Preemptions suffered due to higher
   priority user tasks or system tasks :              51
Worst case blocking time in a single period (us):
                                                     648
Cumulative early deadlines (us):              7681009.00
Context Switches :                                 1283
Delay Expirations :                                1231

============================================================


============================================================

Task Statistics for task : Task 4
------------------------------------------------------------
Cumulative Execution_Time (us):                 1843072
Deadlines Met :                                     616
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              47
Worst case blocking time in a single period (us):
                                                     681
Cumulative early deadlines (us):              6907093.00
Context Switches :                                  663
Delay Expirations :                                 615

============================================================


============================================================

Task Statistics for task : Task 3
------------------------------------------------------------
Cumulative Execution_Time (us):                 1843072
Deadlines Met :                                     308
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              353
Worst case blocking time in a single period (us):
                                                    1333
Cumulative early deadlines (us):              5961515.00
Context Switches :                                  661
Delay Expirations :                                 307

============================================================


============================================================

Task Statistics for task : Task 2
------------------------------------------------------------
```

```
Cumulative Execution_Time (us):                  1843226
Deadlines Met :                                      154
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             355
Worst case blocking time in a single period (us):
                                                    2506
Cumulative early deadlines (us):              5023971.00
Context Switches :                                   509
Delay Expirations :                                  153

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                  1843226
Deadlines Met :                                       77
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             587
Worst case blocking time in a single period (us):
                                                   10151
Cumulative early deadlines (us):                23088.00
Context Switches :                                   664
Delay Expirations :                                   76

=======================================================


Simulation Time (us):                           10007547
User Cumulative Task Execution Time (us):        9215668
User Deadlines Met :                                2387
User Deadlines Missed :                                0
Context Switches :                                  3780
Delay Expirations :                                 2382
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       217918
System Task Execution Time (us):                   767168
Idle Time (us):                                     24711
Percentage User Task Execution Time :           92.087182
Percentage System Task Execution :               7.665895
Percentage Idle Time :                           0.246924

=============================================================================
```

C.1.4.2 Scheduling Failure

```
                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp2_fail
                     Execution
Task name             Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------            --------- ------------------------  ------------
Task 5                   1496      8122    /   123.12         8122
   Rendezvous        : none


Task 4                   2992     16244    /    61.56        16244
   Rendezvous        : none


Task 3                   5984     32488    /    30.78        32488
   Rendezvous        : none


Task 2                  11969     64976    /    15.39        64976
   Rendezvous        : none


Task 1                  23938    129952    /     7.70       129952
   Rendezvous        : none


================================================================================


                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit
```

```
Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=======================================================

Task Statistics for task : Task 5
-------------------------------------------------------
Cumulative Execution_Time (us):                1843072
Deadlines Met :                                   1232
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            51
Worst case blocking time in a single period (us):
                                                   616
Cumulative early deadlines (us):            7677033.00
Context Switches :                                1283
Delay Expirations :                               1231

=======================================================


=======================================================

Task Statistics for task : Task 4
-------------------------------------------------------
Cumulative Execution_Time (us):                1843072
Deadlines Met :                                   616
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            48
Worst case blocking time in a single period (us):
                                                   807
Cumulative early deadlines (us):            6904137.00
Context Switches :                                664
Delay Expirations :                               615

=======================================================


=======================================================

Task Statistics for task : Task 3
-------------------------------------------------------
Cumulative Execution_Time (us):                1843072
Deadlines Met :                                   308
Deadlines Missed :                                   0
Preemptions suffered due to higher
```

```
   priority user tasks or system tasks :            355
Worst case blocking time in a single period (us):
                                                    1490
Cumulative early deadlines (us):             5958069.00
Context Switches :                                  663
Delay Expirations :                                 307

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                1843226
Deadlines Met :                                    154
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           358
Worst case blocking time in a single period (us):
                                                    4051
Cumulative early deadlines (us):             4980937.00
Context Switches :                                  512
Delay Expirations :                                 153

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                1843226
Deadlines Met :                                     76
Deadlines Missed :                                   1
First deadline missed at :                     8446880
Execution completed at :                       8479379
Cumulative late deadlines (us):               32499.00
Preemptions suffered due to higher
   priority user tasks or system tasks :           583
Worst case blocking time in a single period (us):
                                                   12721
Cumulative early deadlines (us):              20934.00
Context Switches :                                  659
Delay Expirations :                                  75

=======================================================


Simulation Time (us):                         10006334
User Cumulative Task Execution Time (us):      9215668
```

```
User Deadlines Met :                                2386
User Deadlines Missed :                                1
Context Switches :                                  3780
Delay Expirations :                                 2381
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       216923
System Task Execution Time (us):                   767629
Idle Time (us):                                     23033
Percentage User Task Execution Time :           92.098345
Percentage System Task Execution :               7.671431
Percentage Idle Time :                           0.230184

================================================================================
```

C.1.4.3 Scheduling Failure - Experiment 2(Hartstone Benchmark Task Parameters)

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task           3 - Get from file
4 - Save to file          5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks         9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp2_fail.hart
                      Execution
Task name              Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------             --------- ------------------------  ------------
Task 5                    1496      8175    /   122.32         8175
   Rendezvous        : none


Task 4                    2992     16349    /    61.17        16349
   Rendezvous        : none


Task 3                    5984     32699    /    30.58        32699
   Rendezvous        : none


Task 2                   11968     65397    /    15.29        65397
   Rendezvous        : none
```

```
Task 1                    23936    130794    /     7.65       130794
   Rendezvous        : none


================================================================================


                      --------------------------------
                      |Rate Monotonic Scheduler Model|
                      --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=========================================================

Task Statistics for task : Task 5
---------------------------------------------------------
Cumulative Execution_Time (us):                 1831104
Deadlines Met :                                    1224
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             52
Worst case blocking time in a single period (us):
                                                     652
Cumulative early deadlines (us):              7695710.00
Context Switches :                                 1276
Delay Expirations :                                1223

=========================================================


=========================================================

Task Statistics for task : Task 4
---------------------------------------------------------
Cumulative Execution_Time (us):                 1831104
Deadlines Met :                                    612
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             565
```

```
Worst case blocking time in a single period (us):
                                                    1371
Cumulative early deadlines (us):                6768534.00
Context Switches :                                  1177
Delay Expirations :                                  611

=============================================================


=============================================================

Task Statistics for task : Task 3
-------------------------------------------------------------
Cumulative Execution_Time (us):                   1831104
Deadlines Met :                                       306
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :                352
Worst case blocking time in a single period (us):
                                                    1691
Cumulative early deadlines (us):                5940818.00
Context Switches :                                    658
Delay Expirations :                                   305

=============================================================


=============================================================

Task Statistics for task : Task 2
-------------------------------------------------------------
Cumulative Execution_Time (us):                   1831104
Deadlines Met :                                       153
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :                464
Worst case blocking time in a single period (us):
                                                    5199
Cumulative early deadlines (us):                3485032.00
Context Switches :                                    617
Delay Expirations :                                   152

=============================================================


=============================================================

Task Statistics for task : Task 1
-------------------------------------------------------------
Cumulative Execution_Time (us):                   1743755
Deadlines Met :                                        20
```

```
Deadlines Missed :                                    52
First deadline missed at :                       2615880
Execution completed at :                         2615924
Cumulative late deadlines (us):             10881108.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             572
Worst case blocking time in a single period (us):
                                                   19856
Cumulative early deadlines (us):                18083.00
Context Switches :                                   592
Delay Expirations :                                   20

=======================================================


Simulation Time (us):                            10005702
User Cumulative Task Execution Time (us):         9068171
User Deadlines Met :                                 2315
User Deadlines Missed :                                52
Context Switches :                                   4268
Delay Expirations :                                  2311
Rendezvous executed :                                   0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       185815
System Task Execution Time (us):                  919111
Idle Time (us):                                    18212
Percentage User Task Execution Time :          90.630033
Percentage System Task Execution :              9.185872
Percentage Idle Time :                          0.182016

=============================================================================
```

C.1.5 Hartstone Results - Experiment 3

```
                HARTSTONE BENCHMARK SUMMARY RESULTS


Baseline test:

=============================================================================

Experiment:    EXPERIMENT_3 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 1 characteristics:
```

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	4.00	16	64.00	4.79 %
3	8.00	8	64.00	4.79 %
4	16.00	4	64.00	4.79 %
5	32.00	2	64.00	4.79 %
			320.00	23.94 %

Experiment step size: 4.64 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_3 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 16 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	47	94.00	7.03 %
2	4.00	31	124.00	9.28 %
3	8.00	23	184.00	13.76 %
4	16.00	19	304.00	22.74 %
5	32.00	17	544.00	40.69 %
			1250.00	93.51 %

Experiment step size: 4.64 %

Test 16 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_3 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 17 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	48	96.00	7.18 %
2	4.00	32	128.00	9.58 %
3	8.00	24	192.00	14.36 %
4	16.00	20	320.00	23.94 %
5	32.00	18	576.00	43.09 %
			1312.00	98.14 %

Experiment step size: 4.64 %

Test 17 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)

1	500.000	0	10	10	234.229
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

Final test performed:

==========

Experiment: EXPERIMENT_3 HARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.80

Test 18 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	49	98.00	7.33 %
2	4.00	33	132.00	9.87 %
3	8.00	25	200.00	14.96 %
4	16.00	21	336.00	25.13 %
5	32.00	19	608.00	45.48 %
			1374.00	102.78 %

Experiment step size: 4.64 %

Test 18 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	0	5	15	1195.361
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	62.500	160	0	0	0.000
5	31.250	320	0	0	0.000

==========

```
================================================================================

Benchmark : Hartstone Benchmark, Version 1.0
Compiler  : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target    : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

   Test 16 of Experiment 3 Harmonic

   Raw (non-tasking) benchmark speed in KWIPS: 1336.80

   Full task set:

      Total       Deadlines        Task Set        Total
      Tasks       Per Second      Utilization      KWIPS
        5           62.00          93.51 %       1250.00

   Highest-frequency task:

      Period      Deadlines          Task          Task
      (msec)      Per Second      Utilization      KWIPS
      31.250        32.00          40.69 %        544.00

   Experiment step size:   4.64 %

================================================================================

              END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.1.6 RATESIM Results - Experiment 3

C.1.6.1 Successful Scheduling

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
```

```
================================================================================
Enter file name to get the task set from: exp3_pass
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                 --------- ------------------------  ------------
Task 5                       13390     31250    /    32.00        31250
   Rendezvous          : none


Task 4                       14886     62500    /    16.00        62500
   Rendezvous          : none


Task 3                       17879    125000    /     8.00       125000
   Rendezvous          : none


Task 2                       23863    250000    /     4.00       250000
   Rendezvous          : none


Task 1                       35832    500000    /     2.00       500000
   Rendezvous          : none


================================================================================


                        ---------------------------------
                        |Rate Monotonic Scheduler Model|
                        ---------------------------------

1 - Add task                2 - Remove task           3 - Get from file
4 - Save to file            5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task               8 - Display tasks         9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


========================================================

Task Statistics for task : Task 5
--------------------------------------------------------
Cumulative Execution_Time (us):                  4284800
Deadlines Met :                                      320
Deadlines Missed :                                     0
Preemptions suffered due to higher
```

```
   priority user tasks or system tasks :              96
Worst case blocking time in a single period (us):
                                                     541
Cumulative early deadlines (us):              5586668.00
Context Switches :                                   416
Delay Expirations :                                  319

=====================================================


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                 2381760
Deadlines Met :                                     160
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             50
Worst case blocking time in a single period (us):
                                                     682
Cumulative early deadlines (us):              5380289.00
Context Switches :                                   210
Delay Expirations :                                  159

=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                 1430320
Deadlines Met :                                      80
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             93
Worst case blocking time in a single period (us):
                                                    1260
Cumulative early deadlines (us):              5135262.00
Context Switches :                                   173
Delay Expirations :                                   79

=====================================================


=====================================================

Task Statistics for task : Task 2
-----------------------------------------------------
Cumulative Execution_Time (us):                  954520
```

```
Deadlines Met :                                    40
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          200
Worst case blocking time in a single period (us):
                                                 4994
Cumulative early deadlines (us):           1285730.00
Context Switches :                                240
Delay Expirations :                                39

=================================================

=================================================

Task Statistics for task : Task 1
-------------------------------------------------
Cumulative Execution_Time (us):                716640
Deadlines Met :                                    20
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          100
Worst case blocking time in a single period (us):
                                                11459
Cumulative early deadlines (us):             3307.00
Context Switches :                                120
Delay Expirations :                                19

=================================================

Simulation Time (us):                       10000015
User Cumulative Task Execution Time (us):    9768040
User Deadlines Met :                             620
User Deadlines Missed :                            0
Context Switches :                              1159
Delay Expirations :                              615
Rendezvous executed :                              0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):   40715
System Task Execution Time (us):              228211
Idle Time (us):                                 3764
Percentage Use  Task Execution Time :       97.680253
Percentage Syst.  Task Execution :           2.282107
Percentage Idle Time :                       0.037640

=============================================================================
```

C.1.6.2 Scheduling Failure - Experment 3

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp3_fail
                     Execution
Task name             Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------            --------- ------------------------  ------------
Task 5                  13398     31250    /    32.00        31250
   Rendezvous        : none


Task 4                  14894     62500    /    16.00        62500
   Rendezvous        : none


Task 3                  17886    125000    /     8.00       125000
   Rendezvous        : none


Task 2                  23870    250000    /     4.00       250000
   Rendezvous        : none


Task 1                  35839    500000    /     2.00       500000
   Rendezvous        : none


================================================================================


                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
```

```
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                4287360
Deadlines Met :                                    320
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            96
Worst case blocking time in a single period (us):
                                                   560
Cumulative early deadlines (us):            5581655.00
Context Switches :                                 416
Delay Expirations :                                319

=====================================================


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                2383040
Deadlines Met :                                    160
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            50
Worst case blocking time in a single period (us):
                                                   703
Cumulative early deadlines (us):            5376893.00
Context Switches :                                 210
Delay Expirations :                                159

=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                1430880
Deadlines Met :                                     80
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            93
Worst case blocking time in a single period (us):
                                                  1285
```

```
Cumulative early deadlines (us):              5132864.00
Context Switches :                                  173
Delay Expirations :                                  79

=============.==========================================


============================================================

Task Statistics for task : Task 2
------------------------------------------------------------
Cumulative Execution_Time (us):                   954800
Deadlines Met :                                       40
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             200
Worst case blocking time in a single period (us):
                                                     4958
Cumulative early deadlines (us):              1281429.00
Context Switches :                                    240
Delay Expirations :                                    39

============================================================


============================================================

Task Statistics for task : Task 1
------------------------------------------------------------
Cumulative Execution_Time (us):                   715614
Deadlines Met :                                        1
Deadlines Missed :                                    18
First deadline missed at :                       1000000
Execution completed at :                         1218168
Cumulative late deadlines (us):               4033523.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             137
Worst case blocking time in a single period (us):
                                                    16425
Cumulative early deadlines (us):                    91.00
Context Switches :                                    138
Delay Expirations :                                     1

============================================================


Simulation Time (us):                            10000001
User Cumulative Task Execution Time (us):         9771694
User Deadlines Met :                                  601
User Deadlines Missed :                                18
Context Switches :                                   1159
```

```
Delay Expirations :                                  597
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       43052
System Task Execution Time (us):                  228139
Idle Time (us):                                       96
Percentage User Task Execution Time :          97.716930
Percentage System Task Execution :              2.281390
Percentage Idle Time :                          0.000960

================================================================================
```

C.1.6.3 Scheduling Failure - Experiment 3(Hartstone Benchmark Task Parameters)

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: exp3_fail.hart
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------  ------------
Task 5                     13465     31250    /    32.00        31250
   Rendezvous         : none


Task 4                     14961     62500    /    16.00        62500
   Rendezvous         : none


Task 3                     17953    125000    /     8.00       125000
   Rendezvous         : none


Task 2                     23938    250000    /     4.00       250000
   Rendezvous         : none


Task 1                     35907    500000    /     2.00       500000
   Rendezvous         : none
```

```
================================================================================

                    ---------------------------------
                    |Rate Monotonic Scheduler Model|
                    ---------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


========================================================

Task Statistics for task : Task 5
--------------------------------------------------------
Cumulative Execution_Time (us):                 4322265
Deadlines Met :                                     321
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             96
Worst case blocking time in a single period (us):
                                                    614
Cumulative early deadlines (us):              5571871.00
Context Switches :                                  417
Delay Expirations :                                 320

========================================================


========================================================

Task Statistics for task : Task 4
--------------------------------------------------------
Cumulative Execution_Time (us):                 2405430
Deadlines Met :                                     160
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :             52
Worst case blocking time in a single period (us):
                                                    688
Cumulative early deadlines (us):              5357718.00
Context Switches :                                  212
Delay Expirations :                                 160
```

```
=======================================================


=======================================================

Task Statistics for task : Task 3
-------------------------------------------------------
Cumulative Execution_Time (us):                1436240
Deadlines Met :                                     80
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            92
Worst case blocking time in a single period (us):
                                                   1242
Cumulative early deadlines (us):            5111730.00
Context Switches :                                 172
Delay Expirations :                                 80

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                 957520
Deadlines Met :                                     40
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           238
Worst case blocking time in a single period (us):
                                                   5432
Cumulative early deadlines (us):             715452.00
Context Switches :                                 278
Delay Expirations :                                 40

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 675469
Deadlines Met :                                      0
Deadlines Missed :                                  18
First deadline missed at :                      500000
Execution completed at :                        734944
Cumulative late deadlines (us):             7821724.00
Preemptions suffered due to higher
```

```
   priority user tasks or system tasks :              100
Worst case blocking time in a single period (us):
                                                    16973
Cumulative early deadlines (us):                     0.00
Context Switches :                                    100
Delay Expirations :                                     0

=========================================================


Simulation Time (us):                             10025600
User Cumulative Task Execution Time (us):          9796924
User Deadlines Met :                                   601
User Deadlines Missed :                                 18
Context Switches :                                    1161
Delay Expirations :                                    600
Rendezvous executed :                                    0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):         48005
System Task Execution Time (us):                    228604
Idle Time (us):                                          0
Percentage User Task Execution Time :            97.719079
Percentage System Task Execution :                2.280203
Percentage Idle Time :                            0.000000

=========================================================================
```

C.2 Task Set B - Nonharmonic

C.2.1 Hartstone Results - Experiment 1

```
                HARTSTONE BENCHMARK SUMMARY RESULTS


Baseline test:

=========================================================================

Experiment:    EXPERIMENT_1 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 1 characteristics:

   Task    Frequency    Kilo-Whets    Kilo-Whets    Requested Workload
    No.     (Hertz)     per period    per second       Utilization
     1        2.00          32          64.00            4.79 %
```

2	2.30	16	36.80	2.75 %
3	4.59	8	36.72	2.75 %
4	6.89	4	27.56	2.06 %
5	9.19	2	18.38	1.37 %
			183.46	13.72 %

Experiment step size: 1.03 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_1 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 58 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	2.30	16	36.80	2.75 %
3	4.59	8	36.72	2.75 %
4	6.89	4	27.56	2.06 %
5	401.92	2	803.84	60.13 %
			968.92	72.48 %

Experiment step size: 1.03 %

Test 58 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	2.488	4020	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_1 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 59 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	2.30	16	36.80	2.75 %
3	4.59	8	36.72	2.75 %
4	6.89	4	27.56	2.06 %
5	408.81	2	817.62	61.16 %
			982.70	73.51 %

Experiment step size: 1.03 %

Test 59 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000

4	145.138	69	0	0	0.000
5	2.446	4087	1	1	0.112

==========

Final test performed:

==========

Experiment: EXPERIMENT_1 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 64 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	2.30	16	36.80	2.75 %
3	4.59	8	36.72	2.75 %
4	6.89	4	27.56	2.06 %
5	443.26	2	886.52	66.32 %
			1051.60	78.67 %

Experiment step size: 1.03 %

Test 64 results:

Test duration (seconds): 10.0

Task No	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	8	6	6	72.856
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	2.256	4377	28	28	0.078

==========

==========

Benchmark : Hartstone Benchmark, Version 1.0

```
Compiler  : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target    : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

   Test 58 of Experiment 1 Nonharmonic

   Raw (non-tasking) benchmark speed in KWIPS: 1336.75

   Full task set:

      Total       Deadlines        Task Set        Total
      Tasks       Per Second      Utilization      KWIPS
        5           417.70          72.48 %        968.92

   Highest-frequency task:

      Period      Deadlines          Task          Task
      (msec)      Per Second      Utilization      KWIPS
       2.488        401.92          60.13 %        803.84

   Experiment step size:   1.03 %

==========================================================================

              END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.2.2 RATESIM Results - Experiment 1

C.2.2.1 Successful Scheduling

```
                        --------------------------------
                        |Rate Monotonic Scheduler Model|
                        --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp1.pas
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
```

```
---------              --------- ------------------------  ------------
Task 5                     1496      2488    /   401.93         2488
   Rendezvous         : none

Task 4                     2992    145138    /     6.89       145138
   Rendezvous         : none

Task 3                     5984    217865    /     4.59       217865
   Rendezvous         : none

Task 2                    11969    434783    /     2.30       434783
   Rendezvous         : none

Task 1                    23938    500000    /     2.00       500000
   Rendezvous         : none

================================================================================


                      ----------------------------
                      |Rate Monotonic Scheduler Model|
                      ----------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


========================================================

Task Statistics for task : Task 5
--------------------------------------------------------
Cumulative Execution_Time (us):                  6012771
Deadlines Met :                                     4019
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             262
Worst case blocking time in a single period (us):
                                                       975
Cumulative early deadlines (us):              2322451.00
```

```
Context Switches :                                    4281
Delay Expirations :                                   4019

================================================================


================================================================

Task Statistics for task : Task 4
----------------------------------------------------------------
Cumulative Execution_Time (us):                     206448
Deadlines Met :                                         69
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               411
Worst case blocking time in a single period (us):
                                                       4245
Cumulative early deadlines (us):                 8913869.00
Context Switches :                                      480
Delay Expirations :                                      68

================================================================


================================================================

Task Statistics for task : Task 3
----------------------------------------------------------------
Cumulative Execution_Time (us):                     275264
Deadlines Met :                                         46
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               539
Worst case blocking time in a single period (us):
                                                       9153
Cumulative early deadlines (us):                 8365900.00
Context Switches :                                      585
Delay Expirations :                                      45

================================================================


================================================================

Task Statistics for task : Task 2
----------------------------------------------------------------
Cumulative Execution_Time (us):                     275287
Deadlines Met :                                         23
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               554
```

```
Worst case blocking time in a single period (us):
                                                    21147
Cumulative early deadlines (us):                7610084.00
Context Switches :                                    577
Delay Expirations :                                    22

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                    478760
Deadlines Met :                                        20
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              964
Worst case blocking time in a single period (us):
                                                    50186
Cumulative early deadlines (us):                6233848.00
Context Switches :                                    984
Delay Expirations :                                    19

=======================================================


Simulation Time (us):                           10000048
User Cumulative Task Execution Time (us):        7248530
User Deadlines Met :                                4177
User Deadlines Missed :                                0
Context Switches :                                  6907
Delay Expirations :                                 4173
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       379394
System Task Execution Time (us):                 1693010
Idle Time (us):                                  1058508
Percentage User Task Execution Time :          72.484952
Percentage System Task Execution :             16.930019
Percentage Idle Time :                         10.585029

=============================================================================
```

C.2.2.2 Scheduling Failure- Experiment 1

```
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp1.fal
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                 --------- ------------------------  ------------
Task 5                        1496      2487    /   402.09         2487
   Rendezvous         : none


Task 4                        2992    145138    /     6.89       145138
   Rendezvous         : none


Task 3                        5984    217865    /     4.59       217865
   Rendezvous         : none


Task 2                       11969    434783    /     2.30       434783
   Rendezvous         : none


Task 1                       23938    500000    /     2.00       500000
   Rendezvous         : none


================================================================================


                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n
```

```
=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                6015416
Deadlines Met :                                   4018
Deadlines Missed :                                   3
First deadline missed at :                     1308162
Execution completed at :                       1308231
Cumulative late deadlines (us):                 118.00
Preemptions suffered due to higher
   priority user tasks or system tasks :            261
Worst case blocking time in a single period (us):
                                                  1060
Cumulative early deadlines (us):            2317596.00
Context Switches :                                4279
Delay Expirations :                               4017

=====================================================


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                 206448
Deadlines Met :                                     69
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            406
Worst case blocking time in a single period (us):
                                                  4280
Cumulative early deadlines (us):            8936035.00
Context Switches :                                 475
Delay Expirations :                                 68

=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                 275264
Deadlines Met :                                     46
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            548
Worst case blocking time in a single period (us):
                                                  9335
Cumulative early deadlines (us):            8366943.00
```

```
Context Switches :                                    594
Delay Expirations :                                    45

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                    275287
Deadlines Met :                                        23
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              556
Worst case blocking time in a single period (us):
                                                     21837
Cumulative early deadlines (us):                7596566.00
Context Switches :                                     579
Delay Expirations :                                     23

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                    478760
Deadlines Met :                                        20
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              950
Worst case blocking time in a single period (us):
                                                     50186
Cumulative early deadlines (us):                6238280.00
Context Switches :                                     970
Delay Expirations :                                     20

=======================================================


Simulation Time (us):                            10000301
User Cumulative Task Execution Time (us):         7251175
User Deadlines Met :                                 4176
User Deadlines Missed :                                 3
Context Switches :                                   6894
Delay Expirations :                                  4173
Rendezvous executed :                                   0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       380368
```

```
System Task Execution Time (us):                1691228
Idle Time (us):                                 1057886
Percentage User Task Execution Time :          72.509567
Percentage System Task Execution :             16.911771
Percentage Idle Time :                         10.578542

================================================================================
```

C.2.2.3 Scheduling Failure - Experiment 1(Hartstone Benchmark Task Parameters)

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task           3 - Get from file
4 - Save to file          5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks         9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp1_fal
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                 --------- ------------------------  ------------
Task 5                        1496      2446    /   408.83         2446
   Rendezvous          : none


Task 4                        2992    145138    /     6.89       145138
   Rendezvous          : none


Task 3                        5984    217865    /     4.59       217865
   Rendezvous          : none


Task 2                       11969    434783    /     2.30       434783
   Rendezvous          : none


Task 1                       23938    500000    /     2.00       500000
   Rendezvous          : none


================================================================================
```

```
                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


==========================================================

Task Statistics for task : Task 5
----------------------------------------------------------
Cumulative Execution_Time (us):                  6116110
Deadlines Met :                                     4087
Deadlines Missed :                                     1
First deadline missed at :                       4500640
Execution completed at :                         4500670
Cumulative late deadlines (us):                    30.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             287
Worst case blocking time in a single period (us):
                                                      980
Cumulative early deadlines (us):              2125119.00
Context Switches :                                  4374
Delay Expirations :                                 4087

==========================================================


==========================================================

Task Statistics for task : Task 4
----------------------------------------------------------
Cumulative Execution_Time (us):                   206448
Deadlines Met :                                       69
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             442
Worst case blocking time in a single period (us):
                                                     4270
Cumulative early deadlines (us):              8862446.00
Context Switches :                                   511
Delay Expirations :                                   68
```

```
=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                275264
Deadlines Met :                                    46
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :           587
Worst case blocking time in a single period (us):
                                                  9784
Cumulative early deadlines (us):            8238883.00
Context Switches :                                633
Delay Expirations :                                45

=====================================================


=====================================================

Task Statistics for task : Task 2
-----------------------------------------------------
Cumulative Execution_Time (us):                275287
Deadlines Met :                                    23
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :           591
Worst case blocking time in a single period (us):
                                                 23153
Cumulative early deadlines (us):            7451840.00
Context Switches :                                614
Delay Expirations :                                22

=====================================================


=====================================================

Task Statistics for task : Task 1
-----------------------------------------------------
Cumulative Execution_Time (us):                478760
Deadlines Met :                                    20
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1044
Worst case blocking time in a single period (us):
                                                 54150
Cumulative early deadlines (us):            5819981.00
```

```
Context Switches :                                    1064
Delay Expirations :                                     19

=======================================================


Simulation Time (us):                              10000129
User Cumulative Task Execution Time (us):           7351869
User Deadlines Met :                                   4245
User Deadlines Missed :                                   1
Context Switches :                                     7195
Delay Expirations :                                    4241
Rendezvous executed :                                     0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):         447283
System Task Execution Time (us):                    1746579
Idle Time (us):                                      901677
Percentage User Task Execution Time :             73.517742
Percentage System Task Execution :                17.465565
Percentage Idle Time :                             9.016654

=============================================================================
```

C.2.3 Hartstone Results - Experiment 2

```
                HARTSTONE BENCHMARK SUMMARY RESULTS


Baseline test:

=============================================================================

Experiment:    EXPERIMENT_2 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 1 characteristics:

   Task    Frequency    Kilo-Whets    Kilo-Whets    Requested Workload
    No.     (Hertz)     per period    per second       Utilization
     1        2.00          32          64.00            4.79 %
     2        2.30          16          36.80            2.75 %
     3        4.59           8          36.72            2.75 %
     4        6.89           4          27.56            2.06 %
     5        9.19           2          18.38            1.37 %
                                       -------          --------
                                       183.46           13.72 %
```

Experiment step size: 1.37 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_2
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 53 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	12.40	32	396.80	29.68 %
2	14.26	16	228.16	17.07 %
3	28.46	8	227.66	17.03 %
4	42.72	4	170.87	12.78 %
5	56.98	2	113.96	8.52 %
			1137.45	85.09 %

Experiment step size: 1.37 %

Test 53 results:

Test duration (seconds): 10.0

Task	Period	Met	Missed	Skipped	Average

No.	in msecs	Deadlines	Deadlines	Deadlines	Late (msec)
1	80.645	125	0	0	0.000
2	70.126	143	0	0	0.000
3	35.139	285	0	0	0.000
4	23.409	428	0	0	0.000
5	17.551	570	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_2 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 52 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	12.20	32	390.40	29.21 %
2	14.03	16	224.48	16.79 %
3	28.00	8	223.99	16.76 %
4	42.03	4	168.12	12.58 %
5	56.06	2	112.12	8.39 %
			1119.11	83.72 %

Experiment step size: 1.37 %

Test 52 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	81.967	121	1	1	7.751
2	71.276	141	0	0	0.000
3	35.716	280	0	0	0.000
4	23.793	421	0	0	0.000
5	17.838	561	0	0	0.000

==========

Final test performed:

==========

Experiment: EXPERIMENT_2 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 57 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	13.20	32	422.40	31.60 %
2	15.18	16	242.88	18.17 %
3	30.29	8	242.35	18.13 %
4	45.47	4	181.90	13.61 %
5	60.65	2	121.31	9.07 %
			1210.84	90.58 %

Experiment step size: 1.37 %

Test 57 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	75.758	72	30	30	11.485
2	65.876	152	0	0	0.000
3	33.010	303	0	0	0.000
4	21.991	455	0	0	0.000
5	16.487	607	0	0	0.000

==========

==========

Benchmark : Hartstone Benchmark, Version 1.0
Compiler : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

```
   Test 53 of Experiment 2 Nonharmonic

   Raw (non-tasking) benchmark speed in KWIPS: 1336.75

   Full task set:

      Total       Deadlines       Task Set       Total
      Tasks       Per Second     Utilization     KWIPS
        5          154.81          85.09 %      1137.45

   Highest-frequency task:

      Period      Deadlines         Task         Task
      (msec)      Per Second     Utilization     KWIPS
      17.551         56.98          8.52 %       113.96

   Experiment step size:   1.37 %

================================================================================

                END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.2.4 RATESIM Results - Experiment 2

C.2.4.1 Successful Scheduling

```
                        -------------------------------
                        |Rate Monotonic Scheduler Model|
                        -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp2.pass
                         Execution
Task name                 Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                --------- ------------------------  ------------
Task 5                      1496     17135    /    58.36        17135
   Rendezvous         : none
```

```
Task 4                    2992     22852    /    43.76        22852
   Rendezvous         : none

Task 3                    5984     34305    /    29.15        34305
   Rendezvous         : none

Task 2                   11969     68446    /    14.61        68446
   Rendezvous         : none

Task 1                   23938     78740    /    12.70        78740
   Rendezvous         : none

================================================================================

                      ----------------------------
                      |Rate Monotonic Scheduler Model|
                      ----------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                873664
Deadlines Met :                                   584
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :          189
Worst case blocking time in a single period (us):
                                                  1267
Cumulative early deadlines (us):            8852982.00
Context Switches :                                773
Delay Expirations :                               583

=====================================================
```

```
=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                1310496
Deadlines Met :                                    438
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            97
Worst case blocking time in a single period (us):
                                                  1218
Cumulative early deadlines (us):            8375303.00
Context Switches :                                 535
Delay Expirations :                                437

=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                1747328
Deadlines Met :                                    292
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           166
Worst case blocking time in a single period (us):
                                                  1886
Cumulative early deadlines (us):            7638389.00
Context Switches :                                 458
Delay Expirations :                                291

=====================================================


=====================================================

Task Statistics for task : Task 2
-----------------------------------------------------
Cumulative Execution_Time (us):                1752421
Deadlines Met :                                    146
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           359
Worst case blocking time in a single period (us):
                                                  3217
Cumulative early deadlines (us):            6481774.00
Context Switches :                                 505
```

```
Delay Expirations :                                    146

================================================================

================================================================

Task Statistics for task : Task 1
----------------------------------------------------------------
Cumulative Execution_Time (us):                    3040126
Deadlines Met :                                        127
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                 721
Worst case blocking time in a single period (us):
                                                        5363
Cumulative early deadlines (us):                2111842.00
Context Switches :                                     848
Delay Expirations :                                    126

================================================================


Simulation Time (us):                             10000091
User Cumulative Task Execution Time (us):          8724035
User Deadlines Met :                                  1587
User Deadlines Missed :                                  0
Context Switches :                                    3119
Delay Expirations :                                   1583
Rendezvous executed :                                    0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):        127663
System Task Execution Time (us):                    716478
Idle Time (us):                                     559578
Percentage User Task Execution Time :             87.239556
Percentage System Task Execution :                 7.164715
Percentage Idle Time :                             5.595729

================================================================================
```

C.2.4.2 Scheduling Failure- Experiment 2

```
                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
```

```
4 - Save to file          5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks         9 - Quit

Enter choice: p
================================================================================
Enter file name to get the task set from: nh_exp2.fail
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------  ------------
Task 5                     1496     17068    /    58.59        17068
   Rendezvous         : none


Task 4                     2992     22763    /    43.93        22763
   Rendezvous         : none


Task 3                     5984     34165    /    29.27        34165
   Rendezvous         : none


Task 2                    11969     68166    /    14.67        68166
   Rendezvous         : none


Task 1                    23938     78431    /    12.75        78431
   Rendezvous         : none


================================================================================


                      -------------------------------
                      |Rate Monotonic Scheduler Model|
                      -------------------------------

1 - Add task              2 - Remove task           3 - Get from file
4 - Save to file          5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks         9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


==================================================

Task Statistics for task : Task 5
--------------------------------------------------
```

```
Cumulative Execution_Time (us):                    876656
Deadlines Met :                                       586
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              195
Worst case blocking time in a single period (us):
                                                      1288
Cumulative early deadlines (us):                8836499.00
Context Switches :                                    781
Delay Expirations :                                   585

=========================================================


=========================================================

Task Statistics for task : Task 4
---------------------------------------------------------
Cumulative Execution_Time (us):                   1316480
Deadlines Met :                                       440
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              114
Worst case blocking time in a single period (us):
                                                      1375
Cumulative early deadlines (us):                8361433.00
Context Switches :                                    554
Delay Expirations :                                   439

=========================================================


=========================================================

Task Statistics for task : Task 3
---------------------------------------------------------
Cumulative Execution_Time (us):                   1753312
Deadlines Met :                                       293
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              109
Worst case blocking time in a single period (us):
                                                      1943
Cumulative early deadlines (us):                7719816.00
Context Switches :                                    402
Delay Expirations :                                   292

=========================================================


=========================================================
```

```
Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                 1759443
Deadlines Met :                                     147
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              306
Worst case blocking time in a single period (us):
                                                   3216
Cumulative early deadlines (us):             6603602.00
Context Switches :                                  453
Delay Expirations :                                 146

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 3057874
Deadlines Met :                                     126
Deadlines Missed :                                    1
First deadline missed at :                      2196068
Execution completed at :                        2207724
Cumulative late deadlines (us):                11656.00
Preemptions suffered due to higher
   priority user tasks or system tasks :              804
Worst case blocking time in a single period (us):
                                                   6827
Cumulative early deadlines (us):             2010660.00
Context Switches :                                  930
Delay Expirations :                                 126

=======================================================


Simulation Time (us):                          10001962
User Cumulative Task Execution Time (us):       8763765
User Deadlines Met :                               1592
User Deadlines Missed :                               1
Context Switches :                                 3119
Delay Expirations :                                1588
Rendezvous executed :                                 0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):      139227
System Task Execution Time (us):                716653
Idle Time (us):                                 521540
Percentage User Task Execution Time :         87.620459
Percentage System Task Execution :             7.165124
```

```
Percentage Idle Time :                    5.214377

================================================================================
```

C.2.4.3 Scheduling Failure - Experiment 2(Hartstone Benchmark Task Parameters)

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task            2 - Remove task          3 - Get from file
4 - Save to file        5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task           8 - Display tasks        9 - Quit

Enter choice:
================================================================================
Enter file name to get the task set from: nh_exp2_fail.hart
                     Execution
Task name             Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------            --------- ------------------------  ------------
Task 5                   1496     17838    /    56.06        17838
   Rendezvous        : none


Task 4                   2992     23793    /    42.03        23793
   Rendezvous        : none


Task 3                   5984     35716    /    28.00        35716
   Rendezvous        : none


Task 2                  11969     71276    /    14.03        71276
   Rendezvous        : none


Task 1                  23938     81967    /    12.20        81967
   Rendezvous        : none


================================================================================

                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------
```

```
1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


============================================================

Task Statistics for task : Task 5
------------------------------------------------------------
Cumulative Execution_Time (us):                    839256
Deadlines Met :                                       561
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              140
Worst case blocking time in a single period (us):
                                                      1070
Cumulative early deadlines (us):                8899834.00
Context Switches :                                    701
Delay Expirations :                                   560

============================================================


============================================================

Task Statistics for task : Task 4
------------------------------------------------------------
Cumulative Execution_Time (us):                   1259632
Deadlines Met :                                       421
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :              126
Worst case blocking time in a single period (us):
                                                      1224
Cumulative early deadlines (us):                8410000.00
Context Switches :                                    547
Delay Expirations :                                   420

============================================================


============================================================

Task Statistics for task : Task 3
------------------------------------------------------------
```

```
Cumulative Execution_Time (us):                    1675520
Deadlines Met :                                        280
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                98
Worst case blocking time in a single period (us):
                                                      1769
Cumulative early deadlines (us):                7804659.00
Context Switches :                                     378
Delay Expirations :                                    279

=========================================================


=========================================================

Task Statistics for task : Task 2
---------------------------------------------------------
Cumulative Execution_Time (us):                    1687629
Deadlines Met :                                        141
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                435
Worst case blocking time in a single period (us):
                                                      3231
Cumulative early deadlines (us):                6536897.00
Context Switches :                                     576
Delay Expirations :                                    140

=========================================================


=========================================================

Task Statistics for task : Task 1
---------------------------------------------------------
Cumulative Execution_Time (us):                    2920436
Deadlines Met :                                        122
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                636
Worst case blocking time in a single period (us):
                                                      5384
Cumulative early deadlines (us):                2799953.00
Context Switches :                                     758
Delay Expirations :                                    121

=========================================================


Simulation Time (us):                             10000065
```

```
User Cumulative Task Execution Time (us):       8382473
User Deadlines Met :                               1525
User Deadlines Missed :                               0
Context Switches :                                 2960
Delay Expirations :                                1520
Rendezvous executed :                                 0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):      132134
System Task Execution Time (us):                  683235
Idle Time (us):                                   934357
Percentage User Task Execution Time :          83.824185
Percentage System Task Execution :              6.832306
Percentage Idle Time :                          9.343509

================================================================================
```

C.2.5 Hartstone Results - Experiment 3

```
                    HARTSTONE BENCHMARK SUMMARY RESULTS

Baseline test:

================================================================================

Experiment:    EXPERIMENT_3 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 1 characteristics:

   Task    Frequency    Kilo-Whets    Kilo-Whets    Requested Workload
    No.     (Hertz)     per period    per second       Utilization
     1        2.00          32          64.00            4.79 %
     2        2.30          16          36.80            2.75 %
     3        4.59           8          36.72            2.75 %
     4        6.89           4          27.56            2.06 %
     5        9.19           2          18.38            1.37 %
                                       -------          --------
                                       183.46           13.72 %

Experiment step size:   1.87 %

--------------------------------------------------------------------------------

Test 1 results:
```

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_3 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 45 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	76	152.00	11.37 %
2	2.30	60	138.00	10.32 %
3	4.59	52	238.68	17.86 %
4	6.89	48	330.72	24.74 %
5	9.19	46	422.74	31.62 %
			1282.14	95.92 %

Experiment step size: 1.87 %

Test 45 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_3 NONHARMONIC
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 46 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	77	154.00	11.52 %
2	2.30	61	140.30	10.50 %
3	4.59	53	243.27	18.20 %
4	6.89	49	337.61	25.26 %
5	9.19	47	431.93	32.31 %
			1307.11	97.78 %

Experiment step size: 1.87 %

Test 46 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	2	9	9	153.293
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

Final test performed:

==========

Experiment: EXPERIMENT_3

Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 48 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	79	158.00	11.82 %
2	2.30	63	144.90	10.84 %
3	4.59	55	252.45	18.89 %
4	6.89	51	351.39	26.29 %
5	9.19	49	450.31	33.69 %
			1357.05	101.52 %

Experiment step size: 1.87 %

Test 48 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	2	9	9	197.211
2	434.783	23	0	0	0.000
3	217.865	46	0	0	0.000
4	145.138	69	0	0	0.000
5	108.814	92	0	0	0.000

==========

==========

Benchmark : Hartstone Benchmark, Version 1.0
Compiler : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

Test 45 of Experiment 3 Nonharmonic

Raw (non-tasking) benchmark speed in KWIPS: 1336.73

Full task set:

```
   Total       Deadlines        Task Set        Total
   Tasks       Per Second      Utilization      KWIPS
     5           24.97          95.92 %       1282.14

  Highest-frequency task:

   Period      Deadlines         Task          Task
   (msec)      Per Second      Utilization      KWIPS
  108.814         9.19          31.62 %        422.74

  Experiment step size:   1.87 %

================================================================================

                END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.2.6 RATESIM Results - Experiment 3

C.2.6.1 Successful Scheduling

```
                        --------------------------------
                        |Rate Monotonic Scheduler Model|
                        --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp3.pas
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------  ------------
Task 5                    34636    108814    /     9.19       108814
   Rendezvous         : none


Task 4                    36132    145138    /     6.89       145138
   Rendezvous         : none


Task 3                    39125    217865    /     4.59       217865
```

```
   Rendezvous        : none

Task 2                   45109    434783    /     2.30       434783
   Rendezvous        : none

Task 1                   57079    500000    /     2.00       500000
   Rendezvous        : none

================================================================================

                      -------------------------------
                      |Rate Monotonic Scheduler Model|
                      -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=======================================================

Task Statistics for task : Task 5
-------------------------------------------------------
Cumulative Execution_Time (us):                 3186512
Deadlines Met :                                      92
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            147
Worst case blocking time in a single period (us):
                                                    1451
Cumulative early deadlines (us):              6754726.00
Context Switches :                                  239
Delay Expirations :                                  91

=======================================================


=======================================================

Task Statistics for task : Task 4
-------------------------------------------------------
```

```
Cumulative Execution_Time (us):                    2493108
Deadlines Met :                                         69
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                91
Worst case blocking time in a single period (us):
                                                       1508
Cumulative early deadlines (us):                 5895415.00
Context Switches :                                      160
Delay Expirations :                                      68

================================================================


================================================================

Task Statistics for task : Task 3
----------------------------------------------------------------
Cumulative Execution_Time (us):                    1799750
Deadlines Met :                                         46
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                96
Worst case blocking time in a single period (us):
                                                       2778
Cumulative early deadlines (us):                 3681325.00
Context Switches :                                      142
Delay Expirations :                                      45

================================================================


================================================================

Task Statistics for task : Task 2
----------------------------------------------------------------
Cumulative Execution_Time (us):                    1037507
Deadlines Met :                                         23
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :                93
Worst case blocking time in a single period (us):
                                                       6506
Cumulative early deadlines (us):                 1342289.00
Context Switches :                                      116
Delay Expirations :                                      22

================================================================


================================================================
```

```
Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                  1141580
Deadlines Met :                                       20
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :               28
Worst case blocking time in a single period (us):
                                                     6517
Cumulative early deadlines (us):              4647639.00
Context Switches :                                    48
Delay Expirations :                                   19

=======================================================


Simulation Time (us):                          10000102
User Cumulative Task Execution Time (us):       9658457
User Deadlines Met :                                250
User Deadlines Missed :                               0
Context Switches :                                  705
Delay Expirations :                                 245
Rendezvous executed :                                 0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):        19272
System Task Execution Time (us):                  147460
Idle Time (us):                                   194185
Percentage User Task Execution Time :          96.583585
Percentage System Task Execution :              1.474585
Percentage Idle Time :                          1.941830

================================================================================
```

C.2.6.2 Scheduling Failure- Experiment 3

```
                    --------------------------------
                    |Rate Monotonic Scheduler Model|
                    --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp3.fal
```

```
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------  ------------
Task 5                     34711    108814    /     9.19       108814
   Rendezvous         : none

Task 4                     36207    145138    /     6.89       145138
   Rendezvous         : none

Task 3                     39200    217865    /     4.59       217865
   Rendezvous         : none

Task 2                     45184    434783    /     2.30       434783
   Rendezvous         : none

Task 1                     57154    500000    /     2.00       500000
   Rendezvous         : none

================================================================================


                      -------------------------------
                      |Rate Monotonic Scheduler Model|
                      -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                3193412
Deadlines Met :                                     92
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           148
Worst case blocking time in a single period (us):
```

```
                                                    1414
Cumulative early deadlines (us):                6748095.00
Context Switches :                                   240
Delay Expirations :                                   91

=======================================================


=======================================================

Task Statistics for task : Task 4
-------------------------------------------------------
Cumulative Execution_Time (us):                   2498283
Deadlines Met :                                        69
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :               88
Worst case blocking time in a single period (us):
                                                     1513
Cumulative early deadlines (us):                5887428.00
Context Switches :                                   157
Delay Expirations :                                   68

=======================================================


=======================================================

Task Statistics for task : Task 3
-------------------------------------------------------
Cumulative Execution_Time (us):                   1803200
Deadlines Met :                                        46
Deadlines Missed :                                      0
Preemptions suffered due to higher
   priority user tasks or system tasks :               97
Worst case blocking time in a single period (us):
                                                     2809
Cumulative early deadlines (us):                3594934.00
Context Switches :                                   143
Delay Expirations :                                   45

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                   1039232
Deadlines Met :                                        23
Deadlines Missed :                                      0
```

```
Preemptions suffered due to higher
   priority user tasks or system tasks :              90
Worst case blocking time in a single period (us):
                                                     6502
Cumulative early deadlines (us):              1325530.00
Context Switches :                                  113
Delay Expirations :                                  22

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                  1143080
Deadlines Met :                                       15
Deadlines Missed :                                     5
First deadline missed at :                       1500000
Execution completed at :                         1681646
Cumulative late deadlines (us):                603912.00
Preemptions suffered due to higher
   priority user tasks or system tasks :              38
Worst case blocking time in a single period (us):
                                                    10833
Cumulative early deadlines (us):               3349588.00
Context Switches :                                    53
Delay Expirations :                                   14

=======================================================


Simulation Time (us):                           10000032
User Cumulative Task Execution Time (us):        9677207
User Deadlines Met :                                 245
User Deadlines Missed :                                5
Context Switches :                                   701
Delay Expirations :                                  240
Rendezvous executed :                                  0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       19561
System Task Execution Time (us):                 146069
Idle Time (us):                                  176736
Percentage User Task Execution Time :         96.771760
Percentage System Task Execution :             1.460685
Percentage Idle Time :                         1.767354

=============================================================================
```

C.2.6.3 Scheduling Failure - Experiment 3(Hartstone Benchmark Task Parameters)

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: nh_exp3.fal
                     Execution
Task name             Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------            --------- ------------------------  ------------
Task 5                 35160    108814    /     9.19       108814
   Rendezvous        : none


Task 4                 36657    145138    /     6.89       145138
   Rendezvous        : none


Task 3                 39649    217865    /     4.59       217865
   Rendezvous        : none


Task 2                 45634    434783    /     2.30       434783
   Rendezvous        : none


Task 1                 57603    500000    /     2.00       500000
   Rendezvous        : none


================================================================================


                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
```

```
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                3234720
Deadlines Met :                                     92
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           144
Worst case blocking time in a single period (us):
                                                  1504
Cumulative early deadlines (us):             6708094.00
Context Switches :                                 236
Delay Expirations :                                 91

=====================================================


=====================================================

Task Statistics for task : Task 4
-------- --------------------------------------------
Cumulative Execution_Time (us):                2529333
Deadlines Met :                                     69
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :            86
Worst case blocking time in a single period (us):
                                                  1535
Cumulative early deadlines (us):             5837444.00
Context Switches :                                 155
Delay Expirations :                                 68

=====================================================


=====================================================

Task Statistics for task : Task 3
-----------------------------------------------------
Cumulative Execution_Time (us):                1823854
Deadlines Met :                                     46
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :           110
Worst case blocking time in a single period (us):
```

```
                                                    2742
Cumulative early deadlines (us):                2705467.00
Context Switches :                                   156
Delay Expirations :                                   45

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                 1049582
Deadlines Met :                                      23
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              75
Worst case blocking time in a single period (us):
                                                    6279
Cumulative early deadlines (us):                1225310.00
Context Switches :                                   98
Delay Expirations :                                  22

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 1152060
Deadlines Met :                                       4
Deadlines Missed :                                   16
First deadline missed at :                       500000
Execution completed at :                         819573
Cumulative late deadlines (us):                 2686601.00
Preemptions suffered due to higher
   priority user tasks or system tasks :              63
Worst case blocking time in a single period (us):
                                                   11433
Cumulative early deadlines (us):                 131786.00
Context Switches :                                   67
Delay Expirations :                                   3

=======================================================


Simulation Time (us):                           10000073
User Cumulative Task Execution Time (us):        9789549
User Deadlines Met :                                234
User Deadlines Missed :                              16
```

```
Context Switches :                                    696
Delay Expirations :                                   229
Rendezvous executed :                                   0
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):        17487
System Task Execution Time (us):                   143575
Idle Time (us):                                     66885
Percentage User Task Execution Time :           97.894775
Percentage System Task Execution :               1.435740
Percentage Idle Time :                           0.668845

================================================================================
```

C.3 Task Set C - Synchronization

C.3.1 Hartstone Results - Experiment 2 (Task Set 1)

HARTSTONE BENCHMARK SUMMARY RESULTS

Baseline test:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 1
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 1 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	6.00	32	192.00	14.36 %
2	12.00	16	192.00	14.36 %
3	24.00	8	192.00	14.36 %
4	96.00	4	384.00	28.73 %
5	96.00	2	192.00	14.36 %
			1152.00	86.18 %

Experiment step size: 0.86 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	166.667	60	0	0	0.000
2	83.333	120	0	0	0.000
3	41.667	240	0	0	0.000
4	10.417	960	0	0	0.000
5	10.417	960	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 1
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 6 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	6.30	32	201.60	15.08 %
2	12.60	16	201.60	15.08 %
3	25.20	8	201.60	15.08 %
4	100.80	4	403.20	30.16 %
5	100.80	2	201.60	15.08 %
			1209.60	90.49 %

Experiment step size: 0.86 %

Test 6 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	158.730	64	0	0	0.000
2	79.365	127	0	0	0.000
3	39.683	253	0	0	0.000
4	9.921	1009	0	0	0.000
5	9.921	1009	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 1
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.75

Test 7 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	6.36	32	203.52	15.22 %
2	12.72	16	203.52	15.22 %
3	25.44	8	203.52	15.22 %
4	101.76	4	407.04	30.45 %
5	101.76	2	203.52	15.22 %
			1221.12	91.35 %

Experiment step size: 0.86 %

Test 7 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	157.233	0	32	32	55.014
2	78.616	128	0	0	0.000
3	39.308	255	0	0	0.000
4	9.827	1018	0	0	0.000
5	9.827	1018	0	0	0.000

==========

Final test performed:
See preceding summary of test 7

==========

Benchmark : Hartstone Benchmark, Version 1.0

```
Compiler  : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target    : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

   Test 6 of Experiment 2 Synchronization Task Set 1

   Raw (non-tasking) benchmark speed in KWIPS: 1336.75

   Full task set:

      Total       Deadlines        Task Set        Total
      Tasks       Per Second      Utilization      KWIPS
        5           245.70          90.49 %      1209.60

   Highest-frequency task:

      Period      Deadlines          Task          Task
      (msec)      Per Second      Utilization      KWIPS
       9.921        100.80          15.08 %       201.60

   Experiment step size:   0.86 %

================================================================================

                END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.3.2 RATESIM Results - Experiment 2 (Task Set 1)

C.3.2.1 Synchronization Successful Scheduling

```
                        --------------------------------
                        |Rate Monotonic Scheduler Model|
                        --------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: sync1_pass
                         Execution
Task name                 Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
```

```
---------                ---------  ------------------------  ------------
Task 4                      2992      10417    /    96.00         10417
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0          0     ANACCEPT      a


Task 5                      1496      10417    /    96.00         10417
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0          0     A_CALL        a


Task 3                      5984      41667    /    24.00         41667
   Rendezvous        : none


Task 2                     11969      83333    /    12.00         83333
   Rendezvous        : none


Task 1                     23938     166667    /     6.00        166667
   Rendezvous        : none


================================================================================


                        --------------------------------
                        |Rate Monotonic Scheduler Model|
                        --------------------------------

1 - Add task               2 - Remove task          3 - Get from file
4 - Save to file           5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task              8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


========================================================

Task Statistics for task : Task 4
--------------------------------------------------------
```

```
Cumulative Execution_Time (us):                    2872320
Deadlines Met :                                        960
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :              1060
Worst case blocking time in a single period (us):
                                                       1012
Cumulative early deadlines (us):                6372124.00
Context Switches :                                     2980
Delay Expirations :                                     959

=======================================================


=======================================================

Task Statistics for task : Task 5
-------------------------------------------------------
Cumulative Execution_Time (us):                    1436160
Deadlines Met :                                        960
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :              1000
Worst case blocking time in a single period (us):
                                                        934
Cumulative early deadlines (us):                4786364.00
Context Switches :                                     1960
Delay Expirations :                                     959

=======================================================


=======================================================

Task Statistics for task : Task 3
-------------------------------------------------------
Cumulative Execution_Time (us):                    1436160
Deadlines Met :                                        240
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               307
Worst case blocking time in a single period (us):
                                                       2510
Cumulative early deadlines (us):                5880079.00
Context Switches :                                      547
Delay Expirations :                                     239

=======================================================


=======================================================
```

```
Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                 1436280
Deadlines Met :                                     120
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              317
Worst case blocking time in a single period (us):
                                                   5104
Cumulative early deadlines (us):              5133282.00
Context Switches :                                  437
Delay Expirations :                                 119

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 1436280
Deadlines Met :                                      60
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :              452
Worst case blocking time in a single period (us):
                                                  19093
Cumulative early deadlines (us):               470352.00
Context Switches :                                  512
Delay Expirations :                                  59

=======================================================


Simulation Time (us):                          10000015
User Cumulative Task Execution Time (us):       8617200
User Deadlines Met :                               2340
User Deadlines Missed :                               0
Context Switches :                                 5479
Delay Expirations :                                2335
Rendezvous executed :                               960
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):       173643
System Task Execution Time (us):                1095856
Idle Time (us):                                  286959
Percentage User Task Execution Time :          86.171871
Percentage System Task Execution :             10.958544
Percentage Idle Time :                          2.869586

=============================================================================
```

C.3.2.2 Synchronization Scheduling Failure - Experiment 2 (Task Set 1)

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: sync1_fail
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                 --------- ------------------------  ------------
Task 4                       2992      9921    /   100.80         9921
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0          0     ANACCEPT      a



Task 5                       1496      9921    /   100.80         9921
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0          0     A_CALL        a



Task 3                       5984     39683    /    25.20        39683
   Rendezvous         : none


Task 2                      11969     79365    /    12.60        79365
   Rendezvous         : none


Task 1                      23938    158730    /     6.30       158730
   Rendezvous         : none


================================================================================

                    -------------------------------
                    |Rate Monotonic Scheduler Model|
```

```
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                3015936
Deadlines Met :                                   1008
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1105
Worst case blocking time in a single period (us):
                                                  1010
Cumulative early deadlines (us):            6188872.00
Context Switches :                                3121
Delay Expirations :                               1007

=====================================================


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                1507968
Deadlines Met :                                   1008
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1051
Worst case blocking time in a single period (us):
                                                   933
Cumulative early deadlines (us):            4524316.00
Context Switches :                                2059
Delay Expirations :                               1007

=====================================================


=====================================================

Task Statistics for task : Task 3
```

```
-------------------------------------------------------
Cumulative Execution_Time (us):                 1507968
Deadlines Met :                                     252
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            388
Worst case blocking time in a single period (us):
                                                   2622
Cumulative early deadlines (us):             5662068.00
Context Switches :                                  640
Delay Expirations :                                 251

=======================================================


=======================================================

Task Statistics for task : Task 2
-------------------------------------------------------
Cumulative Execution_Time (us):                 1508094
Deadlines Met :                                     126
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            416
Worst case blocking time in a single period (us):
                                                   7616
Cumulative early deadlines (us):             2718723.00
Context Switches :                                  542
Delay Expirations :                                 125

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                 1310681
Deadlines Met :                                       0
Deadlines Missed :                                   54
First deadline missed at :                       158730
Execution completed at :                         224525
Cumulative late deadlines (us):             34108980.00
Preemptions suffered due to higher
   priority user tasks or system tasks :            466
Worst case blocking time in a single period (us):
                                                  27396
Cumulative early deadlines (us):                  0.00
Context Switches :                                  466
Delay Expirations :                                   0
```

```
=======================================================


Simulation Time (us):                          10000100
User Cumulative Task Execution Time (us):       8850647
User Deadlines Met :                               2394
User Deadlines Missed :                              54
Context Switches :                                 5771
Delay Expirations :                                2390
Rendezvous executed :                              1008
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):     182082
System Task Execution Time (us):                1149237
Idle Time (us):                                       0
Percentage User Task Execution Time :         88.505585
Percentage System Task Execution :            11.492255
Percentage Idle Time :                         0.000000

=============================================================================
```

C.3.2.3 Scheduling Failure - Task Set 1(Hartstone Benchmark Task Parameters)

```
                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
=============================================================================
Enter file name to get the task set from: sync1_hart.fail
                      Execution
Task name              Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------             --------- ------------------------  ------------
Task 5                   1496      9827    /   101.76         9827
   Rendezvous :
       Start      Length    Type       Name
       -----      ------    ----       ----
         0          0     A_CALL      a



Task 4                   2992      9827    /   101.76         9827
   Rendezvous :
       Start      Length    Type       Name
```

```
-----      ------    ----      ----
  0          0     ANACCEPT     a


Task 3                  5984     39308    /    25.44        39308
   Rendezvous        : none

Task 2                 11969     78616    /    12.72        78616
   Rendezvous        : none

Task 1                 23938    157233    /     6.36       157233
   Rendezvous        : none

================================================================================

                    -------------------------------
                    |Rate Monotonic Scheduler Model|
                    -------------------------------

1 - Add task             2 - Remove task          3 - Get from file
4 - Save to file         5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task            8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                1522928
Deadlines Met :                                   1018
Deadlines Missed :                                   0
Preemptions suffered due to higher
   priority user tasks or system tasks :          1065
Worst case blocking time in a single period (us):
                                                    955
Cumulative early deadlines (us):            4460447.00
Context Switches :                                2083
Delay Expirations :                               1017

=====================================================
```

```
=========================================================

Task Statistics for task : Task 4
---------------------------------------------------------
Cumulative Execution_Time (us):                  3045856
Deadlines Met :                                     1018
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :            1104
Worst case blocking time in a single period (us):
                                                     879
Cumulative early deadlines (us):              6141945.00
Context Switches :                                  3140
Delay Expirations :                                 1017

=========================================================


=========================================================

Task Statistics for task : Task 3
---------------------------------------------------------
Cumulative Execution_Time (us):                  1525920
Deadlines Met :                                     255
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             294
Worst case blocking time in a single period (us):
                                                    2325
Cumulative early deadlines (us):              5664070.00
Context Switches :                                  549
Delay Expirations :                                 254

=========================================================


=========================================================

Task Statistics for task : Task 2
---------------------------------------------------------
Cumulative Execution_Time (us):                  1522692
Deadlines Met :                                     127
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             419
Worst case blocking time in a single period (us):
                                                    6891
Cumulative early deadlines (us):              2681980.00
Context Switches :                                  546
```

```
Delay Expirations :                                    127

=======================================================


=======================================================

Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                    1259118
Deadlines Met :                                          0
Deadlines Missed :                                      52
First deadline missed at :                          157233
Execution completed at :                            224525
Cumulative late deadlines (us):                 46335528.00
Preemptions suffered due to higher
   priority user tasks or system tasks :                465
Worst case blocking time in a single period (us):
                                                       26016
Cumulative early deadlines (us):                        0.00
Context Switches :                                       465
Delay Expirations :                                        0

=======================================================


Simulation Time (us):                              10004000
User Cumulative Task Execution Time (us):           8876514
User Deadlines Met :                                   2418
User Deadlines Missed :                                  52
Context Switches :                                     5715
Delay Expirations :                                    2415
Rendezvous executed :                                  1018
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):         228590
System Task Execution Time (us):                    1127278
Idle Time (us):                                           0
Percentage User Task Execution Time :                88.729648
Percentage System Task Execution :                   11.268273
Percentage Idle Time :                                0.000000

=============================================================================
```

C.3.3 Hartstone Results - Experiment 2 (Task Set 2)

HARTSTONE BENCHMARK SUMMARY RESULTS

Baseline test:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 2
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 1 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	2.00	32	64.00	4.79 %
2	4.00	16	64.00	4.79 %
3	8.00	8	64.00	4.79 %
4	32.00	4	128.00	9.58 %
5	32.00	0	0.00	0.00 %
			320.00	23.94 %

Experiment step size: 2.39 %

Test 1 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	500.000	20	0	0	0.000
2	250.000	40	0	0	0.000
3	125.000	80	0	0	0.000
4	31.250	320	0	0	0.000
5	31.250	320	0	0	0.000

==========

Last test with no missed/skipped deadlines:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 2
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 28 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	7.40	32	236.80	17.71 %
2	14.80	16	236.80	17.71 %
3	29.60	8	236.80	17.71 %
4	118.40	4	473.60	35.43 %
5	118.40	0	0.00	0.00 %
			1184.00	88.57 %

Experiment step size: 2.39 %

Test 28 results:

Test duration (seconds): 10.0

Task No.	Period in msecs	Met Deadlines	Missed Deadlines	Skipped Deadlines	Average Late (msec)
1	135.135	75	0	0	0.000
2	67.568	149	0	0	0.000
3	33.784	297	0	0	0.000
4	8.446	1185	0	0	0.000
5	8.446	1185	0	0	0.000

==========

Test when deadlines first missed/skipped:

==========

Experiment: EXPERIMENT_2 SYNCHRONIZATION TASK SET 2
Completion on: Miss/skip 50 deadlines

Raw speed in Kilo-Whetstone Instructions Per Second (KWIPS): 1336.73

Test 29 characteristics:

Task No.	Frequency (Hertz)	Kilo-Whets per period	Kilo-Whets per second	Requested Workload Utilization
1	7.60	32	243.20	18.19 %
2	15.20	16	243.20	18.19 %
3	30.40	8	243.20	18.19 %
4	121.60	4	486.40	36.39 %
5	121.60	0	0.00	0.00 %
			1216.00	90.97 %

```
Experiment step size:   2.39 %

------------------------------------------------------------------------

Test 29 results:

Test duration (seconds):  10.0

   Task     Period        Met         Missed       Skipped       Average
    No.   in msecs    Deadlines     Deadlines    Deadlines    Late (msec)
     1     131.579         0            38           38          53.666
     2      65.789       152             0            0           0.000
     3      32.895       304             0            0           0.000
     4       8.224      1216             0            0           0.000
     5       8.224      1216             0            0           0.000

========================================================================


Final test performed:
   See preceding summary of test 29



========================================================================

Benchmark : Hartstone Benchmark, Version 1.0
Compiler  : System Designers XD Ada MC68020 Ver 1.0, Kernel Ver V1.2A-33
Target    : MVME133A-20 32-bit Monoboard Microcomputer (68020 @ 20.0 MHz)

Characteristics of best test for this experiment:
(no missed/skipped deadlines)

   Test 28 of Experiment 2 Synchronization Task Set 2

   Raw (non-tasking) benchmark speed in KWIPS: 1336.73

   Full task set:

      Total       Deadlines        Task Set        Total
      Tasks       Per Second     Utilization      KWIPS
        5           288.60         88.57 %      1184.00

   Highest-frequency task:

      Period      Deadlines          Task          Task
      (msec)      Per Second     Utilization      KWIPS
       8.446        118.40          0.00 %         0.00
```

```
   Experiment step size:    2.39 %

================================================================================

                END OF HARTSTONE BENCHMARK SUMMARY RESULTS
```

C.3.4 RATESIM Results - Experiment 2 (Task Set 2)

C.3.4.1 Synchronization Successful Scheduling - Experiment 2 (Task Set 2)

```
                        -------------------------------
                        |Rate Monotonic Scheduler Model|
                        -------------------------------

1 - Add task                2 - Remove task           3 - Get from file
4 - Save to file            5 - Perform simulation    6 - Rate Monotonic Theorem
7 - Edit task               8 - Display tasks         9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: sync2_pass
                          Execution
Task name                  Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------                 --------- ------------------------  ------------
Task 4                       2992      8717    /   114.72         8717
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0        1496     ANACCEPT      a



Task 5                          1      8717    /   114.72         8717
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0           0     A_CALL       a



Task 3                       5984     34868    /    28.68        34868
```

```
   Rendezvous        : none

Task 2                   11969     69735    /    14.34        69735
   Rendezvous        : none

Task 1                   23938    139470    /     7.17       139470
   Rendezvous        : none

================================================================================

                        ---------------------------------
                        |Rate Monotonic Scheduler Model|
                        ---------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


========================================================

Task Statistics for task : Task 4
--------------------------------------------------------
Cumulative Execution_Time (us):                  3434816
Deadlines Met :                                     1148
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :            1248
Worst case blocking time in a single period (us):
                                                    1152
Cumulative early deadlines (us):              5637780.00
Context Switches :                                  3544
Delay Expirations :                                 1147

========================================================


========================================================

Task Statistics for task : Task 5
--------------------------------------------------------
```

```
Cumulative Execution_Time (us):                    1148
Deadlines Met :                                    1148
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :           1253
Worst case blocking time in a single period (us):
                                                    912
Cumulative early deadlines (us):             5465088.00
Context Switches :                                 2401
Delay Expirations :                                1147

=========================================================


=========================================================

Task Statistics for task : Task 3
---------------------------------------------------------
Cumulative Execution_Time (us):                 1717408
Deadlines Met :                                     287
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            335
Worst case blocking time in a single period (us):
                                                   2390
Cumulative early deadlines (us):             5941042.00
Context Switches :                                  622
Delay Expirations :                                 286

=========================================================


=========================================================

Task Statistics for task : Task 2
---------------------------------------------------------
Cumulative Execution_Time (us):                 1723536
Deadlines Met :                                     144
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            348
Worst case blocking time in a single period (us):
                                                   5028
Cumulative early deadlines (us):             5076117.00
Context Switches :                                  492
Delay Expirations :                                 143

=========================================================


=========================================================
```

```
Task Statistics for task : Task 1
-------------------------------------------------------
Cumulative Execution_Time (us):                  1713056
Deadlines Met :                                       71
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             550
Worst case blocking time in a single period (us):
                                                    18999
Cumulative early deadlines (us):                111951.00
Context Switches :                                    621
Delay Expirations :                                    71

=======================================================


Simulation Time (us):                           10007179
User Cumulative Task Execution Time (us):        8589964
User Deadlines Met :                                2798
User Deadlines Missed :                                0
Context Switches :                                  6532
Delay Expirations :                                 2794
Rendezvous executed :                               1148
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):      237238
System Task Execution Time (us):                 1302097
Idle Time (us):                                   115118
Percentage User Task Execution Time :           85.838017
Percentage System Task Execution :              13.011629
Percentage Idle Time :                           1.150354

=======================================================================
```

C.3.4.2 Synchronization Scheduling Failure - Experiment 2 (Task Set 2)

```
                    ------------------------------
                    |Rate Monotonic Scheduler Model|
                    ------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
=================== ============================================================
Enter file name to get the task set from: sync2_fail
```

```
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               ---------  ------------------------  ------------
Task 4                     2992      8705    /   114.88         8705
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0         1496    ANACCEPT      a


Task 5                        1      8705    /   114.88         8705
   Rendezvous :
       Start      Length     Type       Name
       -----      ------     ----       ----
         0            0    A_CALL        a


Task 3                     5984     34819    /    28.72        34819
   Rendezvous        : none


Task 2                    11969     69638    /    14.36        69638
   Rendezvous        : none


Task 1                    23938    139276    /     7.18       139276
   Rendezvous        : none


================================================================================


                      ---------------------------------
                      |Rate Monotonic Scheduler Model|
                      ---------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: p
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


==================================================
```

```
Task Statistics for task : Task 4
-------------------------------------------------------
Cumulative Execution_Time (us):                 3437808
Deadlines Met :                                    1149
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :           1259
Worst case blocking time in a single period (us):
                                                    1051
Cumulative early deadlines (us):              5629713.00
Context Switches :                                 3557
Delay Expirations :                                1148

=======================================================


=======================================================

Task Statistics for task : Task 5
-------------------------------------------------------
Cumulative Execution_Time (us):                    1149
Deadlines Met :                                    1149
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :           1275
Worst case blocking time in a single period (us):
                                                     934
Cumulative early deadlines (us):              5456871.00
Context Switches :                                 2424
Delay Expirations :                                1148

=======================================================


=======================================================

Task Statistics for task : Task 3
-------------------------------------------------------
Cumulative Execution_Time (us):                 1721972
Deadlines Met :                                    287
Deadlines Missed :                                    0
Preemptions suffered due to higher
   priority user tasks or system tasks :            421
Worst case blocking time in a single period (us):
                                                    2819
Cumulative early deadlines (us):              5896343.00
Context Switches :                                 708
Delay Expirations :                                287

=======================================================
```

```
=========================================================

Task Statistics for task : Task 2
---------------------------------------------------------
Cumulative Execution_Time (us):                  1723536
Deadlines Met :                                      144
Deadlines Missed :                                     0
Preemptions suffered due to higher
   priority user tasks or system tasks :             341
Worst case blocking time in a single period (us):
                                                    7522
Cumulative early deadlines (us):              4878305.00
Context Switches :                                   485
Delay Expirations :                                  143

=========================================================


=========================================================

Task Statistics for task : Task 1
---------------------------------------------------------
Cumulative Execution_Time (us):                  1712001
Deadlines Met :                                       69
Deadlines Missed :                                     2
First deadline missed at :                       8078008
Execution completed at :                         8078013
Cumulative late deadlines (us):               49502.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             537
Worst case blocking time in a single period (us):
                                                   27126
Cumulative early deadlines (us):              70667.00
Context Switches :                                   606
Delay Expirations :                                   69

=========================================================


Simulation Time (us):                           10002048
User Cumulative Task Execution Time (us):        8596466
User Deadlines Met :                                2798
User Deadlines Missed :                                2
Context Switches :                                  6629
Delay Expirations :                                 2795
Rendezvous executed :                               1149
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):      237328
System Task Execution Time (us):                 1331845
Idle Time (us):                                    73729
```

```
Percentage User Task Execution Time :          85.947058
Percentage System Task Execution :             13.315723
Percentage Idle Time :                          0.737139

================================================================================
```

C.3.4.3 Scheduling Failure - Task Set 2(Hartstone Benchmark Task Parameters)

```
                        -------------------------------
                        |Rate Monotonic Scheduler Model|
                        -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice: g
================================================================================
Enter file name to get the task set from: sync2_hart.fail
                        Execution
Task name                Time(us)  Period(us)/Frequency(Hz)  Deadline(us)
---------               --------- ------------------------  ------------
Task 5                        1      8224    /   121.60         8224
   Rendezvous :
       Start      Length    Type       Name
       -----      ------    ----       ----
         0          0     A_CALL      a



Task 4                     2992      8224    /   121.60         8224
   Rendezvous :
       Start      Length    Type       Name
       -----      ------    ----       ----
         0         1496    ANACCEPT     a



Task 3                     5984     32895    /    30.40        32895
   Rendezvous        : none


Task 2                    11969     65789    /    15.20        65789
   Rendezvous        : none


Task 1                    23938    131579    /     7.60       131579
```

```
   Rendezvous         : none


================================================================================


                      -------------------------------
                      |Rate Monotonic Scheduler Model|
                      -------------------------------

1 - Add task              2 - Remove task          3 - Get from file
4 - Save to file          5 - Perform simulation   6 - Rate Monotonic Theorem
7 - Edit task             8 - Display tasks        9 - Quit

Enter choice:
================================================================================
Enter length of simulation in microseconds: 10_000_000
Print the Event History (y or n) : n


=====================================================

Task Statistics for task : Task 5
-----------------------------------------------------
Cumulative Execution_Time (us):                  1216
Deadlines Met :                                  1216
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :         1299
Worst case blocking time in a single period (us):
                                                  926
Cumulative early deadlines (us):           5191538.00
Context Switches :                               2515
Delay Expirations :                              1215

=====================================================


=====================================================

Task Statistics for task : Task 4
-----------------------------------------------------
Cumulative Execution_Time (us):                3638272
Deadlines Met :                                  1216
Deadlines Missed :                                  0
Preemptions suffered due to higher
   priority user tasks or system tasks :         1330
Worst case blocking time in a single period (us):
                                                 1040
Cumulative early deadlines (us):           5374430.00
```

```
Context Switches :                                    3762
Delay Expirations :                                   1215

=========================================================


=========================================================

Task Statistics for task : Task 3
---------------------------------------------------------
Cumulative Execution_Time (us):                    1819136
Deadlines Met :                                        304
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               480
Worst case blocking time in a single period (us):
                                                      2855
Cumulative early deadlines (us):                5650640.00
Context Switches :                                     784
Delay Expirations :                                    303

=========================================================


=========================================================

Task Statistics for task : Task 2
---------------------------------------------------------
Cumulative Execution_Time (us):                    1819288
Deadlines Met :                                        152
Deadlines Missed :                                       0
Preemptions suffered due to higher
   priority user tasks or system tasks :               534
Worst case blocking time in a single period (us):
                                                      8965
Cumulative early deadlines (us):                2538453.00
Context Switches :                                     686
Delay Expirations :                                    151

=========================================================


=========================================================

Task Statistics for task : Task 1
---------------------------------------------------------
Cumulative Execution_Time (us):                    1311486
Deadlines Met :                                          0
Deadlines Missed :                                      54
First deadline missed at :                          131579
Execution completed at :                            194282
```

```
Cumulative late deadlines (us):              71648499.00
Preemptions suffered due to higher
   priority user tasks or system tasks :             539
Worst case blocking time in a single period (us):
                                                   29457
Cumulative early deadlines (us):                    0.00
Context Switches :                                   539
Delay Expirations :                                    0

=====================================================


Simulation Time (us):                          10000040
User Cumulative Task Execution Time (us):       8589398
User Deadlines Met :                               2888
User Deadlines Missed :                              54
Context Switches :                                 7016
Delay Expirations :                                2884
Rendezvous executed :                              1216
Cumulative induced priority inversion
    time due to DELAY statement jitter (us):     250963
System Task Execution Time (us):                1410426
Idle Time (us):                                       0
Percentage User Task Execution Time :         85.893636
Percentage System Task Execution :            14.104204
Percentage Idle Time :                         0.000000

=============================================================================
```

Appendix D. *RATESIM Source Code*

This appendix is available upon request, direct requests to:

Major Paul Bailor
Department of the Air Force
AFIT/ENG WPAFB, OH 45433-7765
email: pbailor@afit.af.mil
comm: (513)255-3708
DSN: 785-3708
or
Captain Rusty Baldwin
210 Blair Drive
Fairborn, OH 45324.

Appendix E. *ACEC Test Results*

This appendix is available upon request, direct requests to:

Major Paul Bailor
Department of the Air Force
AFIT/ENG WPAFB, OH 45433-7765
email: pbailor@afit.af.mil
comm: (513)255-3708
DSN: 785-3708
or
Captain Rusty Baldwin
210 Blair Drive
Fairborn, OH 45324.

Appendix F. *RATESIM User's Manual*

This appendix is available upon request, direct requests to:

Major Paul Bailor
Department of the Air Force
AFIT/ENG WPAFB, OH 45433-7765
email: pbailor@afit.af.mil
comm: (513)255-3708
DSN: 785-3708
or
Captain Rusty Baldwin
210 Blair Drive
Fairborn, OH 45324.

Bibliography

1. Altman, Neal. *Hartstone: Synthetic Benchmark Requirements for Hard Real-Time Applications*. Technical Report CMU/SEI-89-TR-23, ESD-89-TR-31, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, June 1989.
2. Bailor, Paul. CSCE 693 Course Notes, 1992. Air Force Institute of Technology WPAFB, OH 45433.
3. Baker, T.P. and Alan Shaw. "The Cyclic Executive Model and Ada," *IEEE Proceedings - Real-Time Systems Symposium*, 120–129 (1988).
4. Clapp, Russell M., et al. "Toward Real-Time Performance Benchmarks for Ada," *Communications of the ACM*, *29*(8):760–778 (August 1986).
5. Department of Defense. *Reference Manual for the Ada Programming Language*, June 1983. ANSI/MIL-STD-1815A-1983.
6. Donohoe, Patrick. *A Survey of Real-Time Performance Benchmarks for the Ada Programming Language*. Technical Report CMU/SEI-87-TR-28, ESD-TR-87-191, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, December 1987.
7. Donohoe, Patrick, et al. *Hartstone Benchmark User's Guide Version 1.0*. Technical Report CMU/SEI-90-UG-1, ESD-90-TR-5, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, March 1990.
8. Klein, Mark H. and Thomas Ralya. *An Analysis of Input/Output Paradigms for Real-Time Systems*. Technical Report CMU/SEI-90-TR-19, ESD-90-TR-220, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, July 1990.
9. Lehoczky, John, et al. "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," *Proceedings IEEE Real-Time Symposium*, 166–171 (1989).
10. Levi, Shem-Tov and Ashok K. Agrawala. *Real Time System Design*. McGraw-Hill, Inc., 1990.
11. Liu, C.L. and J.W. Layland. "Scheduling Algorithms for Multi-Programming in a Hard-Real-Time Environment," *Journal of the Association for Computing Machinery*, *20*(1):46–61 (January 1973).
12. Locke, C. Douglass. "Scheduling in Real Time," *Unix Review*, *8*(9):48–54 (Sep 1990).
13. Mossakowski, Mr. Telecon 2 Nov 92.
14. Motorola, Inc. *MVME133A-20 VMEmodule 32-Bit Monoboard Microcomputer User's Manual*. Tempe, Arizona, April 1987. MVME133A/D1.
15. Motorola, Inc. *32-Bit Microprocessor User's Manual* (Third Edition). Tempe, Arizona, 1990. MC68020UM/AD REV 3.
16. Newport, John R. *Avionics System Computer Performance Testing*. Technical Report TR-2437, Indianapolis, Indiana 46219-2189: Naval Avionics Center, February 1989.
17. Product Support Division. *Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide Release 3.0*. Technical Report D500-12565-1, P.O. Box 7730 Wichita, Kansas: Boeing Defense and Space Group, December 1991.
18. Product Support Division. *Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) User's Guide*. Technical Report D500-12564-1, P.O. Box 7730 Wichita, Kansas: Boeing Defense and Space Group, December 1991.

19. Product Support Division. *Ada Compiler Evaluation Capability (ACEC) Version Description Document Release 3.0*. Technical Report D500-12563-1, P.O. Box 7730 Wichita, Kansas: Boeing Defense and Space Group, December 1991.

20. Roy, Dan, et al. Personal Communications, 1992. Carnegie-Mellon University/Software Engineering Institute.

21. Sha, L., et al. "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," *IEEE Transactions on Computers* (September 1990).

22. Sha, Lui and John B. Goodenough. *Real-Time Scheduling Theory and Ada*. Technical Report CMU/SEI-89-TR-14, ESD-TR-89-22, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, April 1989.

23. Space Systems Division, SSD/MWBX. *Statement of Work - Block 6 Risk Reduction*. P.O. Box 90009, Los Angeles, CA 90009, April 1991.

24. System Designers Software, Inc. *Developing XD Ada Programs on VMS Systems for the MC68020*. Cambridge, Massachusetts, June 1989. DMA-0001A.

25. System Designers Software, Inc. *XD Ada MC68020 Run-Time Reference Manual*. Cambridge, Massachusetts, June 1989. DMA-0004A.

26. Weiderman, Nelson. *Factors Causing Unexpected Variations in Ada Benchmarks*. Technical Report CMU/SEI-87-TR-22, ESD-87-TR-173, Pittsburgh, Pennsylvania 15213: Carnegie-Mellon University/Software Engineering Institute, October 1987.

Vita

Rusty Olen Baldwin was born in Tulsa, Oklahoma on June 13, 1961. He graduated from Manzano High School, Albuquerque, New Mexico in 1979 and enlisted in the United States Air Force in July, 1981. He served as an Instrumentation Mechanic until his acceptance into the Airman Education and Commissioning Program in June, 1985. He entered New Mexico State University, Las Cruces, New Mexico in July, 1985 and graduated with honors in December, 1987 with a Bachelor of Science in Electrical Engineering degree. On April 13, 1988 he received a commission in the United States Air Force. His first assignment was to the Defense Meteorological Satellite Program, System Program Office, Space Systems Division, Los Angeles Air Force Base, California where he directed two major defense contractors' development of a new generation of military weather satellites. He entered the Air Force Institute of Technology in June, 1991.

Permanent address: 210 Blair Drive
Fairborn, Ohio 45324

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
A MODEL FOR DETERMINING TASK SET SCHEDULABILITY IN THE PRESENCE OF SYSTEM EFFECTS

5. FUNDING NUMBERS

6. AUTHOR(S)
Rusty O. Baldwin, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCS/ENG/92D-02

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
WL/AAAF-3, Software Concepts Group,
Avionics Directorate, Wright Laboratory
Wright-Patterson AFB, OH 45433-6583

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This research developed a parameterized model that accounts for system overhead and determines when an Ada runtime environment can no longer successfully execute a given Ada task set and still meet all deadlines. The Ada Compiler Evaluation Capability benchmark was used to characterize an actual runtime environment. Using that data, a generic model of a preemptive, rate monotonic priority based runtime system was developed which accounts for overhead due to clock updates, context switching, task suspension, and synchronization. Validation was based on the Hartstone benchmark. First, the benchmark was executed using the actual runtime environment. Then, those results were compared with the execution of the benchmark using the model. In all cases, except one, the model predicted the point where the task set would fail. A runtime system optimization omitted from model caused the single failure. Experiments conducted using the model allowed the demonstration of the following results. System overhead can be modeled within the existing framework of rate monotonic scheduling theory. Runtime optimizations can be extremely sensitive to phase relationships between task periods and workloads and can render a schedulable task set unschedulable. Requirements of the task set and the performance of the runtime system must be considered simultaneously.

14. SUBJECT TERMS
software engineering, models, real time, scheduling, avionics, rate monotonic, simulation

15. NUMBER OF PAGES
260

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL