VRTX/68000
Timing Reference

Document Number 590092001
December 1983

REV.	REVISION HISTORY	PRINT DATE
—001	First printing; 2.10	12/83

Hunter & Ready, Inc.
445 Sherman Avenue
P.O. Box 60803
Palo Alto, CA 94306-0803
415/326-2950
TELEX:278835

Table of Contents

1. Introduction 1

2. Execution Time Formulas 2

2.1 Shortcuts 4
2.2 Loops 4
2.3 Wait States 4
2.4 Dynamic RAM Refresh 5
2.5 Cases 5
2.6 Reschedule And Timeslice 5
2.7 Hooks 6
2.8 TCB Chain 6
2.9 An Example 7

3. Interrupts Off Formulas 8

4. System Calls 9

VRTX/68000[1] Timing Reference

1 Introduction

Real-time programs are inherently time-critical. If a task does not complete its work on time, <u>the system has failed</u> just as emphatically as if it had produced erroneous data. To build a real-time system that consistently delivers ontime results, the software engineer must be able to precisely characterize the performance of the hardware and software components from which the system is constructed. This is not an easy job because of the multitude of factors: processor type and clock rate, bus and memory design, compiled code quality, and operating system overhead are but a few these factors. Yet only with detailed component performance data can running times be estimated with the accuracy that real-time applications demand.

To aid in this effort, hardware manufacturers customarily supply performance specifications for their components—processors, memories, buses and the like. In marked contrast, software vendors traditionally provide little, if any, timing information. Yet software has no less influence on total system performance. Breaking with tradition, this <u>Timing Reference</u> describes in detail the performance of Hunter & Ready's VRTX/68000 silicon software component. The description aims to answer the two performance questions that are vital to real-time applications.

1. How long does it take to execute a system call?

2. How long are interrupts disabled?

The information contained in this document has many uses; among other things it can help you:

- Compare VRTX's performance to that of other real-time executives (bearing in mind the "apples and oranges" syndrome which complicates direct comparisons).

- Analyze software design alternatives—for example, memory can sometimes be saved by allocating data structures dynamically; however, the cost in increased execution time may make static allocation the better choice in some situations.

- Evaluate hardware design tradeoffs (e.g., whether you will gain more from a faster processor or faster memory).

---

1. The information presented in this document applies to Version 2.10 of VRTX/68000.

- Verify that VRTX and your hardware can field the worst-case interrupt bursts anticipated in your application.

The Timing Reference is primarily organized to facilitate quick calculation of specific performance figures; should you read it front to back you will find considerable repetition. (For a short overview of VRTX/68000's performance, consult VRTX/68000 Timing Summary, Hunter & Ready Document No. 02110.) The material is presented in three sections. The first two sections basically define the terms used in the third. In the third section, each VRTX call is covered in alphabetical order. For each call a set of formulas describes how long the call will take to execute and the maximum interval during which interrupts will be disabled. The formulas cover every set of hardware and software conditions that an application is likely to encounter.

To use this material effectively you should be familiar with the VRTX system calls and should understand microprocessor hardware basics. For example, the term "wait state" should be familiar to you.

2 Execution Time Formulas

To account for the hardware- and software-related variables that can affect the execution time of a system call, each call is described by a formula which yields the total number of clock cycles consumed by the call. (To convert clock cycles to microseconds, divide by the processor clock frequency in megaHertz, e.g., 8 for an 8MHz clock.)

Figure 1 is a flowchart of a hypothetical VRTX system call. How the structure of this call would be reflected in its timing formula is discussed below. First, the formula expressed as you would find it later in this book:

Formula	**a** + **ib**
Terms	**a** = 155 + 23**ram** + 18**rom** **i** = Number of tasks in the system. **b** = 29 + 16**ram** + 8**rom**

Every formula contains at least the **a** term; this term represents the execution time to traverse the "main path" through the code. The main path is the code that is executed one time per call; it includes the time for the TRAP instruction that invokes the call and the RTE instruction that ends it. (High-level language programmers note: the main path does not include interface library parameter processing.)

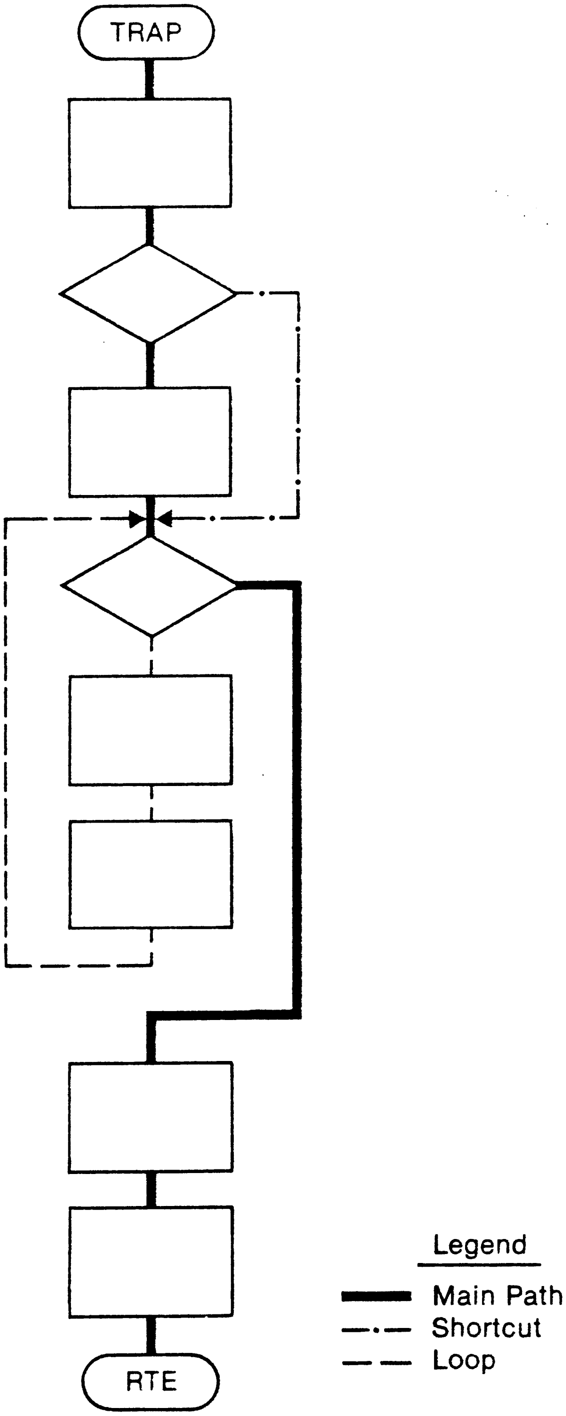

Figure 1. Typical VRTX System Call Flow

2.1 Shortcuts

Notice the "shortcut" in Figure 1. Many system calls contain one or more such branches which, when executed, effectively shorten the length of the main path, thereby slightly decreasing the value of **a.** However, the value quoted for **a** in this document consistently assumes that shortcuts are not taken. Thus, the figures you derive for the **a** term may be slightly conservative in some cases. The effect is minimal, however, since taking even all the shortcuts in a call would not reduce **a**'s value by more than a few percent.

2.2 Loops

The second term in the formula (**ib**) corresponds to the loop in Figure 1; a call may contain zero or more loops. Such a loop is a section of code that may be executed zero or more times (i.e., it is a WHILE loop) in a given invocation of the call. The value of **i** when the call is invoked determines the number of iterations through the loop, while the value of **b** specifies the time it takes to execute the loop once. In our hypothetical call, the loop will be executed once for each task in the system. Multiple loops in a call are denoted by terms of the form **jc**, **kd**, etc.

2.3 Wait States

Both the **a** and **b** terms are themselves defined as sub-formulas consisting of three terms (e.g., **a** = 155 + 23**ram** + 18**rom**). The first term specifies a number of clock cycles assuming 0 wait states[2]. The second and third terms specify the penalties for RAM and ROM wait states respectively, if any; these will be zero in 0-wait state systems. Note that the RAM and ROM referred to here are the code and data areas associated with VRTX, i.e., the locations used for "VRTX system RAM" and the area where the VRTX code itself is located. Wait states incurred by accesses to other memory areas (e.g., to user code or data located on different boards than VRTX) are irrelevant. To use a sub-formula, substitute for **ram** and **rom** the number of wait states incurred in the corresponding access in your system. For example, consider the sub-formula

a = 155 + 23**ram** + 18**rom**

In a 0-wait state system **a** evaluates to 155 (i.e., 155 + 0 + 0). In a system with 1 wait state per VRTX RAM access and 2 wait states per VRTX ROM access, **a** is 214 (i.e., 155 + 23 + 36).

---

2. Do not confuse our use of the term "wait state" with the "states" into which Motorola literature divides the 68000 bus cycle. When we say wait state we mean one full clock period; Motorola bus states are one-half clock period long.

2.4 Dynamic RAM Refresh

The contents of dynamic RAM chips must be periodically refreshed typically every 2-4 milliseconds. If the processor tries to read or write a location that is being refreshed, the refresh logic will insert a wait state into the bus cycle. The effect is to make memory occasionally appear to be slower than it actually is. This effect should be accounted for in the VRTX performance formulas. Unfortunately, the diversity of refresh implementations prevents us from doing so. Nevertheless, if you only need a "ballpark" number, you can add 5% to the RAM clock figures quoted in the formulas to account for refresh-generated wait states. To obtain a more precise figure, you will need to work with your hardware engineers to establish how long refresh takes and how often it occurs. You can then increase the RAM clock figures by either a statistical or a worst-case amount.

2.5 Cases

Often you will find more than one formula given for a system call; this is because the time required to complete the call depends on the context in which it is executed. For example, consider SC_POST, which posts a message to a mailbox. At the time this call is executed one of the following will be true:

1. No task is pended at the target mailbox.

2. A task is pended at the mailbox with a timeout value of 0 (wait indefinitely).

3. A task is pended at the mailbox with a non-zero timeout value.

Each such condition we call a case. Since the processing required to complete the call is substantially case-dependent, we provide a separate formula for each case.

> NOTE: While all cases which affect timing to any significant degree are documented, not <u>every</u> case is described. To do so is infeasible and would yield only marginally useful information. If, however, you identify a case that is crucial to your application, and the case is not described in this document, contact the Hunter and Ready Service and Support Group for assistance.

No formulas are provided for error cases (e.g., an SC_POST to a full mailbox) because such cases should not be encountered in production systems. In any event, a system call executed under erroneous conditions will always complete faster than any valid case.

2.6 Reschedule And Timeslice

VRTX has two subroutines, called "resched" and "tslice," that are executed by many system calls. Basically, resched suspends the executing task and selects the next task for execution; it constitutes VRTX's basic task switching time. Tslice is only called when timeslicing is enabled. Then, whenever a task is suspended, including at the end of a timeslice, tslice is called to move the TCB of the suspended task to the rear of its priority group on the TCB chain (the TCB chain is explained later). These routines have variable execution times which are themselves expressed as formulas. When a system call executes either of these routines, the routine is simply identified as a term in the system call formula. For example, the case of a task suspending itself by calling SC_TSUSPEND, has the following formula:

a + **ib** + **tslice** + **resched.**

The formulas for the tslice and resched routines are defined in the next section before the system calls themselves.

2.7 Hooks

Some formulas contain one of the following terms: **tcrehook, tdelhook** or **tswaphook.** These terms account for the time spent when VRTX calls a user-written routine when a task is created, deleted or swapped (switched), respectively. If no such routine is present (as specified in the Configuration Table), the term evaluates to 0. If a routine is present, the corresponding term is defined with its own sub-formula, in the manner of resched and tslice. The sub-formula itself contains a term (**yourcreatetime, yourdeltime** or **yourswaptime**) for which you substitute the time it takes your routine to run. NOTE: when calculating this time, do not include the RTS instruction that terminates your routine and returns to VRTX; it is already accounted for in term **a** of the main formula.

2.8 TCB Chain

Frequently a term will be defined as the "number of TCBs preceding that of (some) task." What does this mean? VRTX links the task control blocks of all non-dormant tasks into an internal data structure known as the TCB chain. (Dormant TCBs—those that have never been used in an SC_TCREATE system call and those that have been the target of an SC_TDELETE system call—are not on the TCB chain.) The TCBs on the chain are ordered by priority, with the TCB of the highest-priority task at the front of the chain. Equal-priority tasks occupy adjacent positions in the chain; each such string of equal-priority tasks is called a <u>priority group</u>. A newly-created task is always inserted at the front of its priority group, if there is one. Similarly, changing a tasks's priority causes the task's TCB to be inserted at the front of its new priority group, if there is one. Thus, if three tasks have priorities of 220, and these priorities have not been changed, the TCB of the most recently-created task is ahead of the other two, while the oldest task's TCB brings up the rear (ahead, however, of any lower-priority TCBs).

When timeslicing is enabled, VRTX rotates the tasks in a priority group each time a task in the group is suspended. For example, suppose three priority-250 tasks are linked in the chain so that task_b is followed by task_c, with task_a at the back of the priority group. Task_b is executing (task_c and task_a are both ready) and it issues an SC_PEND to an empty mailbox. VRTX inserts task_b behind task_a in the TCB chain and executes task_c which is now at the front of the priority group. When task_c is suspended, VRTX will place it behind task_b, making task_a the next to run.

Therefore, when timeslicing is not in effect, the number of TCBs ahead of a particular task's TCB consists of the following:

- all higher-priority tasks;

- all younger tasks of the same priority;

- all equal-priority tasks whose priority has been changed after the task in question was created or after the task in question's priority was changed.

When timeslicing is enabled, the TCBs of all higher-priority tasks are ahead of the TCB in question. If the TCB is in a priority group, from zero to all-but-one of the TCBs in the priority group can also be ahead of the TCB in question.

2.9 An Example

To help crystallize our explanation of the timing formulas, we now walk through an example. Suppose your hardware is running an 8mHz 68000 with one wait state per ROM access and two wait states per RAM access. You want to know how long it takes to allocate a block of RAM from a VRTX free pool partition (i.e., how long SC_GBLOCK takes). The formula for SC_GBLOCK is given as follows:

Formula	**a** + **ib**
Terms	**a** = 618 + 68**ram** + 73**rom**
	i = Number of partitions created after the target partition was created.
	b = 46 + 3**ram** + 7**rom**

For this example, suppose no SC_PCREATE calls have previously been executed, so that the free pool consists of only the single partition which VRTX creates at initialization-time (i.e., term **i** = 0). Substituting in the formula we get

a = (618 + (68*2) + (73*1)) = 827
i = 0
b = (46 + (3*2) + (7*1)) = 59

a + **ib** = 827 + (0*59) = <u>827 clock cycles</u>.

Dividing by the processor's clock frequency of 8 yields 103.375 microseconds to allocate a block of memory. Note that this figure does not include any clock cycles lost to memory refresh.

3 Interrupts Off Formulas

During the execution of many system calls, VRTX must disable interrupts while updating critical data structures. To see why this is so, consider a situation in which VRTX is in the midst of adding an item to a queue in behalf of a task that issued an SC_QPOST system call. If VRTX allowed interrupts to be recognized during the critical update sequence, occurence of an interrupt could ready a higher priority task which could issue an SC_POST to the same queue. While the consequence of having thus corrupted the queue data structure is unpredictable, it would clearly be unacceptable.

Depending on the call, and on the case (as defined in the last section), VRTX may disable interrupts zero or more times as it executes a system call. What is of interest here is not how many times interrupts are disabled, nor the aggregate time spent with interrupts disabled. It is, rather, the maximum time spent with interrupts off—since this is what contributes to worst-case interrupt latency. Accordingly, the interrupts-off formulas defne the longest period during which VRTX disables interrupts in the execution of a system call.

As in the previous section, the interrupts-off timing is presented as one or more formulas. The structure of these formulas is identical to those presented previously, although the formulas and cases are usually simpler.

To clarify how to use the formulas, consider an example. Suppose you want to know the maximum time interrupts are disabled when a free pool partition is extended. The interrupts-off formula for the SC_PEXTEND system call is quoted below.

Formula	**a** + **ib**
Terms	**a** = 316 + 32**ram** + 38**rom**
	i = Number of blocks already in the target partition.
	b = 36 + 4**ram** + 4**rom**

For purposes of this example, we assume that the hardware is driven by an 8MHz clock and that one wait state is incurred per RAM access and two wait states per ROM access. We also assume that the partition to be extended already has 25 blocks. Substituting in the formula we obtain:

a = (316 + (32*1) + (38*2)) = 424
i = 25
b = (36 + (4*1) + (4*2)) = 48

a + **ib** = 424 + (25*48) = 1624 clock cycles.

Dividing by the processor's clock frequency of 8 yields 203 microseconds during which the system will be unable to respond to an interrupt. Again, this figure will have to be adjusted upward to account for memory refresh.

4 System Calls

The VRTX system calls are listed in this section alphabetically. The resched and tslice routines come first, however, as they are called by other system calls.

<u>Execution</u>

Formula	**a** + **ib** + **tswaphook**
Terms	**a** = 846 + 116**ram** + 84**rom** = 84.6 + 80 = 164.6
	i = Number of <u>suspended</u> TCBs preceding that of the highest-priority ready task.
	b = 56 + 3**ram** + 9**rom** = 5.6 + 9.8 = 15.4
	tswaphook = (54 + 8**ram** + 5 **rom** + **yourswaptime**) if **tswaphook** is specified in the Configuration Table; else **tswaphook** = 0.

* * *

<u>Interrupts Off</u>

Formula	**a**
Terms	**a** = 0

* * *

Execution

Case 1 — The task being changed from executing to ready is the only one in its priority group (i.e., there are no other non-dormant tasks of the same priority).

Formula — **a**

Terms — **a** = 74 + 5**ram** + 11**rom** = 7.4 + 6.4 = 13.8

* * *

Case 2 — Other non-dormant tasks have the same priority as the task being changed from executing to ready.

Formula — **a** + **ib** + **jc**

Terms — **a** = 156 + 13**ram** + 20**rom** = 15.6 + 13.2 = 28.8

i = Number of non-dormant tasks with higher priority than the target task.

b = 36 + 4**ram** + 4**rom** = 3.6 + 5.2 = 8.8

j = Number of non-dormant tasks with the same priority as the target task (not including the target task itself).

c = 56 + 3**ram** + 9**rom**

* * *

Interrupts Off

Case 1 — The task being changed from executing to ready is the only one in its priority group (i.e., there are no other non-dormant tasks of the same priority).

Formula — **a**

Terms — **a** = 74 + 5**ram** + 11**rom** = 7.4 + 6.4 = 13.8

* * *

Case 2 — Other non-dormant tasks have the same priority as the task being changed from executing to ready.

Formula **a** + **ib** + **jc**

Terms

a = 156 + 13**ram** + 20**rom** = 15.6 + 13.2 = 28.8

i = Number of non-dormant tasks with higher priority than the target task.

b = 36 + 4**ram** + 4**rom**

j = Number of non-dormant tasks with the same priority as the target task (not including the target task itself).

c = 56 + 3**ram** + 9**rom**

* * *

Execution

Case 1	No message is available.
Formula	**a**
Terms	**a** = 414 + 45**ram** + 48**rom**

* * *

Case 2	A message is available.
Formula	**a**
Terms	**a** = 428 + 47**ram** + 49**rom**

* * *

Interrupts Off

Case 1	No message is available.
Formula	**a**
Terms	**a** = 200 + 20**ram** + 24**rom**

* * *

Case 2	A message is available.
Formula	**a**
Terms	**a** = 214 + 22**ram** + 25**rom**

* * *

Execution

Formula **a** + **ib**

Terms **a** = 618 + 68**ram** + 73**rom**

i = Number of memory partitions that were created after the partition specified in this call.

b = 46 + 3**ram** + 7**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 290 + 32**ram** + 34**rom**

* * *

Execution

Case 1	A character is available.
Formula	**a**
Terms	**a** = 542 + 56**ram** + 65**rom**

* * *

Case 2	No character is available (GETC buffer is empty).
Formula	**a** + **ib** + **tslice** + **resched**
Terms	**a** = 700 + 71**ram** + 89**rom**
	i = Number of tasks (not including this one) already suspended on an SC_GETC call.
	b = 44 + 4**ram** + 6**rom**
	tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.
	resched = See **Reschedule.**

* * *

Interrupts Off

Case 1	A character is available.
Formula	**a**
Terms	**a** = 328 + 31**ram** + 41**rom**

* * *

Case 2	No character is available (GETC buffer is empty).
Formula	**a** + **ib** + **tslice**
Terms	**a** = 300 + 23**ram** + 46**rom**
	i = Number of tasks (not including calling task) already suspended on an SC_GETC call.

b = 44 + 4**ram** + 6**rom**

tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.

* * *

Execution

Formula	**a**
Terms	**a** = 408 + 45**ram** + 47**rom**

* * *

Interrupts Off

Formula	**a**
Terms	**a** = 198 + 20**ram** + 24**rom**

* * *

Execution

Formula **a**

Terms **a** = 354 + 43**ram** + 39**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 0

* * *

Execution

Formula	**a** + **ib** + **jc**
Terms	**a** = 974 + 109**ram** + 117**rom**
	i = Number of existing partitions.
	b = 46 + 3**ram** + 7**rom**
	j = Number of blocks in partition being created.
	c = 284 + 28**ram** + 35**rom**

* * *

Interrupts Off

Formula	**a**
Terms	**a** = 152 + 15**ram** + 19**rom**

* * *

Execution

Case 1	A message is available.
Formula	**a**
Terms	**a** = 430 + 47**ram** + 50**rom**

* * *

Case 2	No message is available and the call specified a timeout value of 0.
Formula	**a** + **tslice** + **resched**
Terms	**a** = 574 + 60**ram** + 72**rom**
	tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.
	resched = See **Reschedule.**

* * *

Case 3	No message is available and the call specifies a non-zero timeout value.
Formula	**a** + **ib** + **tslice** + **resched**
Terms	**a** = 804 + 81**ram** + 105**rom**
	i = Number of tasks currently suspended because of a timeout (SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, SC_TDELAY) whose time remaining is less than the timeout value specified in the call.
	b = 54 + 3**ram** + 9**rom**
	tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.
	resched = See **Reschedule.**

* * *

Interrupts Off

Case 1 A message is available.

Formula **a**

Terms **a** = 216 + 22**ram** + 26**rom**

* * *

Case 2 No message is available and the call specified a timeout value of 0.

Formula **a** + **tslice**

Terms **a** = 448 + 43**ram** + 60**rom**

tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.

* * *

Case 3 No message is available and the call specifies a non-zero timeout value.

Formula **a** + **ib** + **tslice**

Terms **a** = 678 + 64**ram** + 93**rom**

i = Number of tasks currently suspended because of a timeout (SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, SC_TDELAY) whose time remaining is less than the timeout value specified in the call.

b = 54 + 3**ram** + 9**rom**

tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.

* * *

Execution

Formula **a** + **ib** + **jc** + **kd**

Terms **a** = 804 + 89**ram** + 95**rom**

i = Number of partitions that were created after the partition specified in this call.

b = 46 + 3**ram** + 7**rom**

j = Number of blocks to be added to the target partition.

c = 284 + 28**ram** + 35**rom**

k = Number of blocks already in the target partition.

d = 36 + 4**ram** + 4**rom**

* * *

Interrupts Off

Formula **a** + **ib**

Terms **a** = 316 + 32**ram** + 38**rom**

i = Number of blocks already in the target partition.

b = 36 + 4**ram** + 4**rom**

* * *

Execution

Case 1 No task is suspended on the target mailbox.

Formula **a** + **ib**

Terms **a** = 496 + 49**ram** + 61**rom**

i = Number of non-dormant tasks.

b = 52 + 4**ram** + 7**rom**

* * *

Case 2 The highest-priority task suspended on the target mailbox has timeout value of 0.

Formula **a** + **ib** + **resched**

Terms **a** = 776 + 76**ram** + 98**rom**

i = Number of TCBs preceding that of the highest-priority task suspended on the target mailbox.

b = 52 + 4**ram** + 7**rom**

resched = See **Reschedule.**

* * *

Case 3 The highest-priority task suspended on the target mailbox has a non-zero timeout value.

Formula **a** + **ib** + **jc** + **resched**

a = 942 + 91**ram** + 120**rom**

i = Number of TCBs preceding that of the highest-priority task suspended on the target mailbox.

b = 52 + 4**ram** + 7**rom**

j = Number of tasks currently suspended because of a timeout (due to SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) with less time remaining than highest-priority task suspended on the target mailbox.

c = 48 + 2**ram** + 8**rom**

* * *

Interrupts Off

Case 1 Execution of the system call does not result in rescheduling.

Formula **a**

Terms **a** = 108 + 16**ram** + 9**rom**

* * *

Case 2 Execution of the system call results in rescheduling.

Formula **a** + **tslice**

Terms **a** = 108 + 4**ram** + 20**rom**

tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Case 1	There is room in the PUTC buffer and there is no outstanding uncompleted UI_TXRDY system call.
Formula	**a**
Terms	**a** = 576 + 59**ram** + 69**rom**

* * *

Case 2	The PUTC buffer is full.
Formula	**a** + **ib** + **tslice** + **resched**
Terms	**a** = 700 + 71**ram** + 89**rom**
	i = Number of tasks suspended for an SC_PUTC (not including the calling task).
	b = 44 + 4**ram** + 6**rom**
	tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0. **resched** = See **Reschedule.**

* * *

Interrupts Off

Case 1	There is room in the PUTC buffer and there is no outstanding uncompleted UI_TXRDY system call.
Formula	**a**
Terms	**a** = 362 + 34**ram** + 45**rom**

* * *

Case 2	The PUTC buffer is full.
Formula	**a** + **ib** + **tslice**
Terms	**a** = 300 + 23**ram** + 46**rom**
	i = Number of tasks suspended for an SC_PUTC (not including the calling task).

b = 44 + 4**ram** + 6**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0

* * *

Execution

Case 1 — No message is available.

Formula — **a** + **ib**

Terms — **a** = 542 + 55**ram** + 67**rom**

i = Number of queues that were created after the target queue.

b = 46 + 3**ram** + 7**rom**

* * *

Case 2 — A message is available.

Formula — **a** + **ib**

Terms — **a** = 688 + 62**ram** + 91**rom**

i = Number of queues that were created after the target queue.

b = 46 + 3**ram** + 7**rom**

* * *

Interrupts Off

Case 1 — No message is available.

Formula — **a**

Terms — **a** = 200 + 19**ram** + 25**rom**

* * *

Case 2 — A message is available.

Formula — **a**

Terms — **a** = 346 + 26**ram** + 49**rom**

* * *

Execution

Formula **a** + **ib**

Terms **a** = 808 + 82**ram** + 104**rom**

i = Number of queues that have been created (not including the one about to be created).

b = 46 + 3**ram** + 7**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 152 + 15**ram** + 19**rom**

* * *

Execution

Case 1	A message is available.
Formula	**a** + **ib**
Terms	**a** = 674 + 62**ram** + 88**rom**
	i = Number of queues created after the target queue.
	b = 46 + 3**ram** + 7**rom**

* * *

Case 2	No message is available and the call specifies a timeout value of 0.
Formula	**a** + **ib** + **tslice** + **resched**
Terms	**a** = 784 + 91**ram** + 92**rom**
	i = Number of queues created after the target queue.
	b = 46 + 3**ram** + 7**rom**
	tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.
	resched = See **Reschedule.**

* * *

Case 3	No message is available and the call specifies a non-zero timeout value.
Formula	**a** + **ib** + **jc** + **tslice** + **resched**
Terms	**a** = 1014 + 112**ram** + 125**rom**
	i = Number of queues created after the target queue.
	b = 46 + 3**ram** + 7**rom**

j = Number of tasks currently suspended because of a timeout (due to SC_PEND with timeout, SC_QPEND with timeout, or SC_TDELAY) whose time remaining is less than the timeout value specified in the call.

c = 54 + 3**ram** + 9**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

resched = See **Reschedule.**

* * *

Interrupts Off

Case 1	A message is available.
Formula	**a**
Terms	**a** = 332 + 26**ram** + 46**rom**

* * *

Case 2	No message is available and the call specifies a timeout value of 0.
Formula	**a** + **tslice**
Terms	**a** = 476 + 44**ram** + 66**rom**
	tslice = See **Timeslice** if timeslicing is enabled, else **tslice** = 0.

* * *

Case 3	No message is available and the call specifies a non-zero timeout value.
Formula	**a** + **ib** + **tslice**
Terms	**a** = 734 + 69**ram** + 102**rom**
	i = Number of tasks currently suspended because of a timeout (due to SC_PEND with timeout, SC_QPEND with timeout, or SC_TDELAY) whose

time remaining is less than the timeout value specified in the call.

b = 54 + 3**ram** + 9**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

* * *

Execution

Case 1	No task is suspended on this queue.
Formula	**a** + **ib**
Terms	**a** = 740 + 66**ram** + 97**rom**
	i = Number of queues that were created after the target queue.
	b = 46 + 3**ram** + 7**rom**

* * *

Case 2	The highest-priority task suspended on the target queue has timeout value of 0.
Formula	**a** + **ib** + **jc** + **resched**
Terms	**a** = 1224 + 104**ram** + 169**rom**
	i = Number of queues that were created after the target queue.
	b = 46 + 3**ram** + 7**rom**
	j = Number of TCBs preceding that of the highest-priority task suspended on the target queue.
	c = 46 + 3**ram** + 7**rom**
	resched = See **Reschedule.**

* * *

Case 3	The highest-priority task suspended on the target queue has a non-zero timeout value.
Formula	**a** + **ib** + **jc** + **kd** + **resched**
	a = 1390 + 119**ram** + 191**rom**
	i = Number of queues that were created after the target queue.

b = 46 + 3**ram** + 7**rom**

j = Number of TCBs preceding that of the highest-priority task suspended on the target queue.

c = 46 + 3**ram** + 7**rom**

k = Number of tasks currently suspended because of a timeout (due to SC_PEND with timeout, SC_QPEND with timeout, or SC_TDELAY) whose time remaining is less than that of the highest-priority task suspended on the target queue.

d = 48 + 2**ram** + 8**rom**

resched = See **Reschedule.**

* * *

Interrupts Off

Case 1 — No task is suspended on this queue.

Formula — **a**

Terms — **a** = 206 + 9**ram** + 34**rom**

* * *

Case 2 — The highest-priority task suspended on this queue is behind the calling task on the TCB chain.

Formula — **a**

Terms — **a** = 108 + 16**ram** + 9**rom**

* * *

Case 3 — The highest-priority task suspended on this queue is ahead of the calling task on the TCB chain.

Formula — **a** + **tslice**

Terms — **a** = 108 + 4**ram** + 20**rom**

tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Formula **a** + **ib** + **jc**

Terms **a** = 584 + 64**ram** + 69**rom**

i = Number of partitions that were created after the target partition.

b = 46 + 3**ram** + 7**rom**

j = Number of allocated blocks in the target partition.

c = 56 + 4**ram** + 8**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 246 + 28**ram** + 28**rom**

* * *

Execution

Formula **a**

Terms **a** = 408 + 45**ram** + 47**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 198 + 20**ram** + 24**rom**

* * *

Execution

Case 1 ID of task to be created is 0.

Formula **a** + **ib** + **tcrehook** + **resched**

Terms **a** = 1238 + 155**ram** + 136**rom**

i = Number of non-dormant tasks with higher priority than the task being created.

b = 50 + 3**ram** + 8**rom**

tcrehook = (106 + 17**ram** + 9**rom** + **yourcreatetime**) if task create hook is specified in Configuration Table, else **tcrehook** = 0.

resched = See **Reschedule** if the created task is of equal or greater priority than the calling task; else **resched** = 0.

* * *

Case 2 ID of task to be created is not 0.

Formula **a** + **ib** + **jc** + **tcrehook** + **resched**

Terms **a** = 1266 + 157**ram** + 140**rom**

i = Number of non-dormant tasks not including the one being created.

b = 58 + 3**ram** + 9**rom**

j = Number of non-dormant tasks with higher priority than the task being created.

c = 50 + 3**ram** + 8**rom**

tcrehook = (106 + 17**ram** + 9**rom** + **yourcreatetime**) if task create hook specified in Configuration Table; else **tcrehook** = 0.

resched = See **Reschedule** if the created task is of equal or greater priority than the calling task; else **resched** = 0.

* * *

Interrupts Off

Formula **a** + **ib**

Terms **a** = 82 + 7**ram** + 22**rom**

i = Number of non-dormant tasks with higher priority than the task being created.

b = 50 + 3**ram** + 8**rom**

* * *

Execution

Formula **a** + **ib** + **tslice** + **resched**

Terms **a** = 744 + 75**ram** + 96**rom**

i = Number of tasks currently suspended because of a timeout (due to SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) whose time remaining is less than the time specified in the call.

b = 54 + 3**ram** + 9**rom**

tslice = see **tslice** if timeslicing is enabled; else **tslice** = 0.

resched = See **Reschedule.**

* * *

Interrupts Off

Formula **a** + **tslice**

Terms **a** = 266 + 15**ram** + 44**rom**

tslice = see **tslice** if timeslicing is enabled; else **tslice** = 0.

* * *

Execution

Case 1 Format 3 (delete self).

Formula **a** + **ib** + **tdelhook** + **resched**

Terms **a** = 842 + 84**ram** + 107**rom**

i = Number of TCBs preceding that of the task to be deleted.

b = 52 + 4**ram** + 7**rom**

tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

resched = See **Reschedule.**

* * *

Case 2 Format 2 (delete task with given ID) and target ID is not that of the caller and the target task is not suspended due to an SC_GETC, SC_PUTC, SC_WAITC, SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or an SC_TDELAY call. (The target task could be suspended due to an SC_TSUSPEND call.)

Formula **a** + **ib** + **tdelhook**

Terms **a** = 1080 + 105**ram** + 140**rom**

i = Number of TCBs preceding that of the target task.

b = 50 + 3**ram** + 8**rom**

tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

* * *

Case 3 Format 2 (delete task with given ID) and target task is suspended due to an uncompleted SC_GETC or SC_PUTC call.

Formula	**a** + **ib** + **jc** + **tdelhook**
Terms	**a** = 1276 + 119**ram** + 167**rom**
	i = Number of TCBs preceding that of the target task.
	b = 50 + 3**ram** + 8**rom**
	j = Number of tasks suspended for an SC_GETC or an SC_PUTC before the target task was suspended.
	c = 48 + 2**ram** + 8**rom**
	tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

* * *

Case 4	Format 2 (delete task with given ID) and target task is suspended for an SC_WAITC call.
Formula	**a** + **ib** + **tdelhook**
Terms	**a** = 1208 + 113**ram** + 158**rom**
	i = Number of TCBs preceding that of the target task.
	b = 50 + 3**ram** + 8**rom**
	tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

* * *

Case 5	Format 2 (delete task with given ID) and target task is suspended for an SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or an SC_TDELAY call.
Formula	**a** + **ib** + **jc** + **tdelhook**
Terms	**a** = 1412 + 131**ram** + 187**rom**

i = Number of TCBs preceding that of the target task.

b = 50 + 3**ram** + 8**rom**

j = Number of tasks currently suspended because of a timeout (due to SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) with less time remaining than the target task.

c = 48 + 2**ram** + 8**rom**

tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

* * *

Case 6

Format 1 (delete all tasks in a priority group, possibly including the calling task) and none of the target tasks is suspended for an SC_GETC, SC_PUTC, SC_WAITC, SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or an SC_TDELAY call.

Formula

a + **ib** + **j**(**c**+**tdelhook**) + **resched**

Terms

a = 786 + 73**ram** + 103**rom**

i = Number of non-dormant tasks of higher priority than the target task group.

b = 58 + 3**ram** + 9**rom**

j = Number of tasks in the target priority group.

c = 276 + 24**ram** + 39**rom**

tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

resched = See **Reschedule** if the calling task is in the target priority group; else **resched** = 0.

* * *

Case 7

Format 1 (delete all tasks in a priority group, possibly including the calling task) and one or more of the target tasks is suspended for an SC_GETC, SC_PUTC, SC_WAITC, SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or SC_TDELAY call.

Formula

a + **ib** + **j**(**c** + **tdelhook**) + **d1** + **d2** ... + **dn** + **resched**

Terms

a = 786 + 73**ram** + 103**rom**

i = Number of non-dormant tasks of higher priority than the target task group.

b = 58 + 3**ram** + 9**rom**

j = Number of tasks in the target priority group.

c = 276 + 24**ram** + 39**rom**

tdelhook = (118 + 19**ram** + 10**rom** + **yourdeltime**) if **tdelhook** is specified in Configuration Table; else **tdelhook** = 0.

dn = Time to delete a task in the priority group:

- If task is not suspended for one of above reasons: **dn** = 0.
- If task is suspended for an SC_GETC or SC_PUTC: **dn** = (158 + 10ram + 23**rom**) + **k**(48 + 2**ram** + 8**rom**).
- If task is suspended for an SC_WAITC: **dn** = 90 + 4**ram** + 14**rom**.
- If task is suspended for an SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout), or SC_TDELAY: **dn** = (294 + 22**ram** + 43**rom**) + **l**(48 + 2**ram** + 8**rom**).

  k = Number of tasks suspended for an SC_GETC or SC_PUTC before the target task was suspended.

  l = Number of tasks currently suspended because of a timeout (SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, SC_TDELAY) with less time remaining than the target task.

* * *

Interrupts Off

Case 1	Format 3 (delete self) or Format 2 and ID is that of calling task.
Formula	**a**
Terms	**a** = 194 + 8**ram** + 34**rom**

* * *

Case 2	Format 2 (delete task with given ID) and target ID is not that of the caller and target task is not suspended due to an SC_GETC, SC_PUTC, SC_WAITC, SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or an SC_TDELAY call. (The target task could be suspended due to an SC_TSUSPEND call.)
Formula	**a**
Terms	**a** = 194 + 20**ram** + 23**rom**

* * *

Case 3	Format 2 (delete task with given ID) and target task is suspended due to an uncompleted SC_GETC call.
Formula	**a** + **ib**
Terms	**a** = 280 + 22**ram** + 41**rom**
	i = Number of tasks suspended for an SC_GETC before the target task was suspended.
	b = 48 + 2**ram** + 8**rom**

* * *

Case 4	Format 2 (delete task with given ID) and target task is suspended for an SC_PUTC call.
Formula	**a** + **ib**
Terms	**a** = 338 + 25**ram** + 49**rom**
	i = Number of tasks suspended for an SC_PUTC before the target task was suspended.
	b = 48 + 2**ram** + 8**rom**

* * *

Case 5	Format 2 (delete task with given ID) and target task is suspended for an SC_WAITC call.
Formula	**a**
Terms	**a** = 270 + 19**ram** + 40**rom**

* * *

Case 6	Format 2 (delete task with given ID) and target task is suspended for an SC_PEND (with non-zero timeout), SC_QPEND (with non-zero timeout) or an SC_TDELAY call.
Formula	**a** + **ib**
Terms	**a** = 474 + 37**ram** + 69**rom**
	i = Number of tasks currently suspended because of a timeout (due to SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) with less time remaining than the target task.
	b = 48 + 2**ram** + 8**rom**

* * *

Case 7	Format 1 (delete all tasks in a priority group); calling task is <u>not</u> in the target priority group.

Formula	**a**
Terms	**a** = 108 + 16**ram** + 9**rom**

* * *

Case 8	Format 1 (delete all tasks in a priority group); calling task is in the target priority group.
Formula	**a** + **tslice**
Terms	**a** = 108 + 4**ram** + 20**rom**
	tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Case 1	Format 3 (calling task).
Formula	**a** + **ib**
Terms	**a** = 658 + 69**ram** + 81**rom**
	i = Number of TCBs preceding that of the calling task.
	b = 52 + 4**ram** + 7**rom**

* * *

Case 2	Format 2 (task specified by ID).
Formula	**a** + **ib**
Terms	**a** = 806 + 86**ram** + 99**rom**
	i = Number of TCBs preceding that of the target task.
	b = 50 + 3**ram** + 8**rom**

* * *

Interrupts Off

Formula	**a**
Terms	**a** = 108 + 16**ram** + 9**rom**

* * *

Execution

Case 1 — Format 3 (change priority of calling task).

Formula — **a** + **ib** + **jc** + **resched**

Terms — **a** = 668 + 73**ram** + 80**rom**

i = Number of TCBs preceding that of the calling task.

b = 52 + 4**ram** + 7**rom**

j = Number of tasks with higher priority than the new priority of the calling task.

c = 68 + 4**ram** + 10**rom**

resched = See **Reschedule.**

* * *

Case 2 — Format 2 (change priority of a task specified by ID number).

Formula — **a** + **ib** + **jc** + **resched**

Terms — **a** = 834 + 91**ram** + 100**rom**

i = Number of TCBs preceding that of the target task.

b = 50 + 3**ram** + 8**rom**

j = Number of tasks with higher priority than the new priority of the target task.

c = 68 + 4**ram** + 10**rom**

resched = See **Reschedule.**

* * *

Interrupts Off

Formula — **a** + **ib**

Terms — **a** = 300 + 19**ram** + 47**rom**

i = Number of tasks with higher priority than the new priority of the calling task.

b = 68 + 4**ram** + 10**rom**

* * *

Execution

Case 1 Format 2 (resume task with a specified ID).

Formula **a** + **ib** + **resched**

Terms **a** = 718 + 78**ram** + 86**rom**

i = Number of TCBs preceding that of the target task.

b = 50 + 3**ram** + 8**rom**

resched = See **Reschedule.**

* * *

Case 2 Format 1 (resume all tasks of a specified priority).

Formula **a** + **ib** + **jc** + **resched**

Terms **a** = 604 + 59**ram** + 77**rom**

i = Number of tasks with higher priority than the target task group.

b = 58 + 3**ram** + 9**rom**

j = Number of tasks in the target priority group.

c = 74 + 5**ram** + 11**rom**

* * *

Interrupts Off

Case 1 Execution of the system call does not result in rescheduling.

Formula **a**

Terms **a** = 108 + 16**ram** + 9**rom**

* * *

Case 2 — Execution of the system call results in rescheduling.

Formula — **a** + **tslice**

Terms — **a** = 108 + 4**ram** + 20**rom**

tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Formula **a**

Terms **a** = 434 + 45**ram** + 52**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 108 + 16**ram** + 9**rom**

* * *

Execution

Case 1 — Format 3 (suspend calling task).

Formula — **a** + **ib** + **tslice** + **resched**

Terms — **a** = 808 + 81**ram** + 102**rom**

i = Number of TCBs preceding that of the calling task.

b = 52 + 4**ram** + 7**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

resched = See **Reschedule.**

* * *

Case 2 — Format 2 (suspend task with given ID).

Formula — **a** + **ib** + **tslice** + **resched**

Terms — **a** = 1028 + 103**ram** + 130**rom**

i = Number of TCBs preceding that of the target task.

b = 50 + 3**ram** + 8**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

resched = See **Reschedule** if target task is calling task; else **resched** = 0.

* * *

Case 3 — Format 1 (suspend all tasks in a priority group).

Formula — **a** + **ib** + **jc** + **resched**

Terms — **a** = 726 + 67**ram** + 96**rom**

i = Number of tasks with higher priority than the target priority group.

b = 58 + 3**ram** + 9**rom**

j = Number of tasks in the target priority group (including the caller, if applicable).

c = 74 + 5**ram** + 11**rom**

resched = See **Reschedule** if calling task is in target priority group, else **resched** = 0.

* * *

Interrupts Off

Case 1 Format 3 (suspend calling task).

Formula **a** + **tslice**

Terms **a** = 360 + 23**ram** + 58**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

* * *

Case 2 Format 2 (suspend task with given ID).

Formula **a** + **tslice**

Terms **a** = 464 + 41**ram** + 63**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

* * *

Case 3 Format 1 (suspend all tasks in a priority group).

Formula **a** + **tslice**

Terms **a** = 396 + 26**ram** + 63**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0.

* * *

Execution

Case 1

No timeslice has expired in the interval between execution of the preceding SC_LOCK call and this SC_UNLOCK call. (A timeslice expires when timeslicing is enabled and a UI_TIMER call zeros VRTX's timeslice counter.)

Formula

a + **resched**

Terms

a = 404 + 46**ram** + 46**rom**

resched = See **Reschedule.**

* * *

Case 2

Timeslicing is enabled and a timeslice has expired in the interval between execution of the preceding SC_LOCK call and this SC_UNLOCK call.

Formula

a + **ib** + **jc** + **resched**

Terms

a = 662 + 70**ram** + 79**rom**

i = Number of non-dormant tasks of higher priority than the task whose timeslice has expired.

b = 36 + 4**ram** + 4**rom**

j = Number of non-dormant tasks of same priority as the task whose timeslice has expired (not including that task).

c = 56 + 3**ram** + 4**rom**

resched = See **Reschedule.**

* * *

Interrupts Off

Formula

a + **tslice**

Terms

a = 108 + 4**ram** + 20**rom**

tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Formula **a** + **tslice** + **resched**

Terms **a** = 572 + 61**ram** + 70**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0;

resched = See **Reschedule.**

* * *

Interrupts Off

Formula **a** + **tslice**

Terms **a** = 318 + 20**ram** + 52**rom**

tslice = See **Timeslice** if timeslicing is enabled; else **tslice** = 0

* * *

Execution

Case 1	No timeslice expired during execution of this interrupt handler (including any nested handlers).
Formula	**a** + **resched**
Terms	**a** = 486 + 64**ram** + 48**rom**
	resched = See **Reschedule** if this interrupt results in a task switch (e.g., the handler posts a message to a mailbox where a higher-priority task is pending); else **resched** = 0.

* * *

Case 2	Timeslicing is enabled and a timeslice has expired during the execution of this interrupt handler. (The expiration is caused by execution of a UI_TIMER call that zeros VRTX's internal timeslice counter. The UI_TIMER call may have been issued by this interrupt handler, in which case this is the timer handler, or by a different handler invoked by a nested interrupt.)
Formula	**a** + **ib** + **jc** + **resched**
Terms	**a** = 744 + 88**ram** + 81**rom**
	i = Number of non-dormant tasks of higher priority than task whose timeslice has expired.
	b = 36 + 4**ram** + 4**rom**
	j = Number of non-dormant tasks of same priority as the task whose timeslice has expired (not including that task).
	c = 56 + 3**ram** + 9**rom**
	resched = See **Reschedule.**

* * *

Interrupts Off

Case 1	Execution of the system call does not result in rescheduling.

Formula	**a**
Terms	**a** = 108 + 16**ram** + 9**rom**

* * *

Case 2	Execution of the system call results in rescheduling.
Formula	**a** + **tslice**
Terms	**a** = 108 + 4**ram** + 20**rom**
	tslice = See **Timeslice** if timeslicing is enabled, and during execution of the call a nested UI_TIMER call decrements VRTX's timeslice count to 0; else **tslice** = 0.

* * *

Execution

Case 1	No task is suspended on the target mailbox.
Formula	**a** + **ib**
Terms	**a** = 442 + 47**ram** + 53**rom**
	i = Number of non-dormant tasks.
	b = 52 + 4**ram** + 7**rom**

* * *

Case 2	Highest-priority task suspended on the target mailbox has a timeout value of 0.
Formula	**a** + **ib**
Terms	**a** = 722 + 74**ram** + 90**rom**
	i = Number of TCBs preceding that of the highest-priority task suspended on the target mailbox.
	b = 52 + 4**ram** + 7**rom**

* * *

Case 3	Highest-priority task suspended on the target mailbox has a non-zero timeout value.
Formula	**a** + **ib** + **jc**
Terms	**a** = 888 + 89**ram** + 112**rom**
	i = Number of TCBs preceding the highest-priority task suspended on the target mailbox.
	b = 52 + 4**ram** + 7**rom**
	j = Number of tasks currently suspended because of a timeout (due to SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) with less time remaining than the highest-priority task suspended on the target mailbox.

c = 48 + **2ram** + **8rom**

* * *

<u>Interrupts Off</u>

Formula	**a**
Terms	**a** = 60 + **3ram** + **9rom**

* * *

<u>Execution</u>

Case 1	No task is suspended on the target queue.
Formula	**a** + **ib**
Terms	**a** = 686 + 64**ram** + 89**rom**
	i = Number of queues that were created after the target queue.
	b = 46 + 3**ram** + 7**rom**

* * *

Case 2	The highest-priority task suspended on the target queue has a timeout value of 0.
Formula	**a** + **ib** + **jc**
Terms	**a** = 1170 + 102**ram** + 161**rom**
	i = Number of queues that were created after the target queue.
	b = 46 + 3**ram** + 7**rom**
	j = Number of TCBs preceding that of the highest-priority task suspended on the target queue.
	c = 46 + 3**ram** + 7**rom**

* * *

Case 3	The highest-priority task suspended on the target queue has a non-zero timeout value.
Formula	**a** + **ib** + **jc** + **kd**
Terms	**a** = 1336 + 117**ram** + 183**rom**
	i = Number of queues that were created after the target queue.
	b = 46 + 3**ram** + 7**rom**

j = Number of TCBs preceding that of the highest-priority task suspended on the target queue.

c = 46 + 3**ram** + 7**rom**

k = Number of tasks currently suspended because of a timeout (SC_PEND with non-zero timeout, SC_QPEND with non-zero timeout, or SC_TDELAY) with less time remaining than the highest-priority task suspended on the target queue.

d = 48 + 2**ram** + 8**rom**

* * *

Interrupts Off

Case 1	No task is suspended on the target queue.
Formula	**a**
Terms	**a** = 206 + 9**ram** + 34**rom**

* * *

Case 2	One or more tasks are suspended on the target queue.
Formula	**a**
Terms	**a** = 70 + 3**ram** + 12**rom**

* * *

Execution

Case 1	The GETC buffer is not full and no task is suspended for an SC_GETC.
Formula	**a**
Terms	**a** = 576 + 61**ram** + 70**rom**

* * *

Case 2	The GETC buffer is empty and one or more tasks are suspended for SC_GETC's.
Formula	**a**
Terms	**a** = 706 + 76**ram** + 85**rom**

* * *

Case 3	The received character is the one for which a task suspended by an SC_WAITC is waiting.
Formula	**a**
Terms	**a** = 538 + 59**ram** + 65**rom**

* * *

Interrupts Off

Formula	**a**
Terms	**a** = 0

* * *

Execution

Formula **a** + **ib**

Terms **a** = 644 + 64**ram** + 82**rom**

i = Number of tasks currently suspended because of a timeout (due to SC_PEND with timeout, SC_QPEND with timeout, or SC_TDELAY) with only one tick remaining.

b = 284 + 22**ram** + 42**rom**

* * *

Interrupts Off

Formula **a**

Terms **a** = 56 + 2**ram** + 10**rom**

* * *

Execution

Case 1	The PUTC buffer is full and one or more tasks are suspended for SC_PUTCs.
Formula	**a**
Terms	**a** = 690 + 78**ram** + 82**rom**

* * *

Case 2	The PUTC buffer has a character available and no tasks are suspended for SC_PUTCs.
Formula	**a**
Terms	**a** = 526 + 61**ram** + 61**rom**

* * *

Interrupts Off

Formula	**a**
Terms	**a** = 0

* * *

