Using the Time Manager

The **Time Manager** is automatically initialized when the system starts up. At that time, the queue of **Time Manager** task records is empty. The Operating System and applications may place records into the queue. Because the delay time for a given task can be as small as 20 microseconds, you need to install an element into the **Time Manager** queue before actually issuing a request to execute it at some future time. You place elements into the queue by calling the **InsTime** procedure or (if you need the fixed-frequency services of the extended **Time Manager**) the **InsXTime** procedure. To activate the request, call **PrimeTime**. The **Time Manager** then marks the specified task record as active by setting the high-order bit in the qType field of that record.

The tmAddr field of the **Time Manager** task record contains the address of a task that the **Time Manager** calls when the time delay specified by a previous call to **PrimeTime** has elapsed. The task can perform any desired actions, so long as those actions do not call the **Memory Manager** (either directly or indirectly) and do not depend on the validity of handles to unlocked blocks. **Time Manager** tasks must also preserve all registers other than A0-A3 and D0-D3.

If the routine specified in the **Time Manager** task record is loaded into the application's heap, then the application must still be active when the specified delay elapses, or the application should call **RmvTime** before it terminates. Otherwise, the **Time Manager** will not know that the address of that routine is not valid when the routine is called. The **Time Manager** will then attempt to call the task, but with a stale pointer. If you want to let the application terminate after it has installed and activated a **Time Manager** task record, you should load the routine into the system heap. Generally, however, you should avoid loading routines into the system heap.

> **Assembly-language note:** In the revised and extended Time Managers, when a Time Manager task is called, register A1 contains a pointer to the Time Manager task record associated with that routine.

There are two ways in which an active queue element can become inactive. First, the specified time delay can elapse, in which case the routine pointed to by the tmAddr field is called. Second, your application can call the **RmvTime** procedure, in which case the amount of time remaining before the delay would have elapsed (the unused time) is reported in the tmCount field of the task record. This feature allows you to use the **Time Manager** to compute elapsed times (see Computing Elapsed Time), which is useful for obtaining performance measurements. Calling **RmvTime** removes an element from the queue whether or not that task is active at the time **RmvTime** is called.

To use the **Time Manager** to perform actions periodically, you simply need to have the routine pointed to by tmAddr call **PrimeTime** again. This technique is illustrated in **‘‘**Performing Periodic Tasks**’’**. Similarly, you can set up a **Time Manager** task to execute a specific number of times by keeping a count of the number of times the task has been called. In cases where the task needs access to your application's global variables (such as a count variable), you need to ensure that the A5 register points to your application's global variables when the task executes and that A5 contains its original value when your task exits. A technique for doing this is illustrated in **Using Application Global Variables in Tasks**.