Computing Elapsed Time

The **RmvTime** procedure in the revised and extended Time Managers returns any unused time in the tmCount field of the task record. This feature makes the **Time Manager** extremely useful for computing elapsed times, which can, in turn, provide performance measurements.

To compute the amount of time that a routine takes to execute, call **PrimeTime** at the beginning of the interval to be measured and specify a delay greater than the expected elapsed time. Then call **RmvTime** at the end of the interval and subtract the unused time returned in tmCount from the original delay passed to **PrimeTime**. To obtain the most accurate results, you should do all timing in microseconds (in which case the tmCount field of the task record has a range of about 35 minutes). To get an exact measurement, you should compute the overhead associated with calling the **Time Manager** and subtract it from the preliminary result. The following program illustrates a technique for computing elapsed time.

```
;allocate and clear a TMTask record on the stack
;setting tmAddr := 0 means no task
        moveq.l     #(tmQSize/2)-1,d0       ;set up loop counter
                        ; to clear TMTask
@clear  clr.w       -(sp)                   ;allocate and clear TMTask record
        dbra        d0,@clear               ;clear it a word at a time
        move.l #60*1000*1000,d7             ;D7 := delay in microseconds
                                            ; (1 minute)
            movea.l     sp,a0               ;A0 points to TMTask
            _InsTime                        ;install the task
            move.l d7,d6                    ;D6 := copy of initial delay
            move.l d7,d0                    ;D0 := delay time
            neg.l       d0                  ;negate it for microseconds
            _PrimeTime                      ;start the timer
            _RmvTime;immediately stop it
;unused time will be returned in negated microseconds,
; so adding is really subtracting
            add.l       tmCount(a0),d7      ;D7 := initial delay -
                                            ; time remaining
;D7 now contains the overhead in microseconds of _PrimeTime and _RmvTime
            movea.l     sp,a0               ;A0 points to TMTask record
            _InsTime                        ;install the task
            move.l d6,d0                    ;D0 := delay time
            neg.l       d0                  ;negate it for microseconds
            _PrimeTime                      ;start the timer
;beginning of code to be timed
; (in this example, a TimeDBRA loop)
            move.w      TimeDBRA,d0         ;number of DBRAs per millisecond
@dbraLoop   dbra        d0,@dbraLoop;waste a millisecond
;end of code to be timed
            _RmvTime;stop the timer
            add.l       tmCount(a0),d6      ;D6 := time used in microseconds
            sub.l       d7,d6               ;subtract the Time Mgr overhead
            adda.w  #tmQSize,sp             ;deallocate TMTask record
;register D6 now contains the number of microseconds
; used by the timed code
```

If you run this code, you might notice that on some models of the Macintosh, register D6 is not very close to 1000 (one millisecond). This is *not* due to a problem in the **Time Manager**. Rather, this occurs because TimeDBRA is the number of DBRA instructions per millisecond when executing out of ROM, and RAM accesses have different timing on some models.

> **Note:** You should not run this sample code on a Macintosh Plus because that computer's ROM does not support the TimeDBRA variable.

You can insert a task record into the **Time Manager**'s queue by calling **InsTime** or **InsXTime**. Use **InsXTime** only if you wish to use the drift-free, fixed-frequency timing services of the extended **Time Manager**; use **InsTime** in all other cases. After you have queued a task record, you can activate it by calling **PrimeTime**. You can remove a task record from the queue by calling **RmvTime**.