T E C H N O T E :
Inside Macintosh: Processes Time Manager Addenda

by Eric Simenel <simenel.e@apple.com>

Apple Developer Technical Support (DTS)

This Note highlights the usage of two fields — `tmReserved` and `tmWakeUp` — that might be unclear after reading the chapter "Time Manager" in *Inside Macintosh: Processes*.

This Note is intended for all developers who want to do time measurement using the Time Manager routines.

Setting Up tmReserved

On page 3-8 of *Inside Macintosh: Processes*, it clearly states that both `tmWakeUp` and `tmReserved` should be set to 0 prior to the first call to `InsXTime` when using the extended Time Manager:

```
theTMTask.tmWakeUp = theTMTask.tmReserved = 0;

InsXTime((QElemPtr)&theTMTask);

PrimeTime((QElemPtr)&theTMTask, 2000);
```

If you do want to do some time measurement, then you have to call `RmvTime` to get the current value of `tmCount`, which leads later to a new call to `InsXTime`, and a call to PrimeTime with a 0 delay which has a special meaning in that case. Although it appears, after much reading, rather clear that you leave the current value of `tmWakeUp` untouched in the `TMTask` structure, you can't be sure what to do about the value of `tmReserved`.

The truth is that prior to October, 1992 (System Software 7.1), you didn't care, but it's more of a concern now, since Apple slightly modified the behavior of the Time Manager to deal with performance issues.

If you leave `tmReserved` untouched, then, after 127 calls to the following code:

```
RmvTime((QElemPtr)&theTMTask);

remaining = theTMTask.tmCount;

InsXTime((QElemPtr)&theTMTask);

PrimeTime((QElemPtr)&theTMTask, 0);
```

for some good but can't-be-disclosed reason, your extended time task is converted into a non-extended time task which, being waked up with a 0 delay PrimeTime (which has no special meaning for a non-extended time task), will suddenly be called and called again — more frequently than it should be.

So, if you perform that kind of time measurement, be sure to write instead:

```
RmvTime((QElemPtr)&theTMTask);

remaining = theTMTask.tmCount;
```

```
theTMTask.tmReserved = 0;

InsXTime((QElemPtr)&theTMTask);

PrimeTime((QElemPtr)&theTMTask, 0);
```

Since the Time Manager, prior to System Software 7.1, doesn't care about `tmReserved`, then you can set `tmReserved` to 0 before each call to `InsXTime` without checking the system version. You still have, of course, to ensure that the Time Manager you're using is the extended one (`gestaltTimeMgrVersion 'tmgr'`, answering `gestaltExtendedTimeMgr` (= 3)), and, at this point, there is no way to tell what's going to happen under Mac OS 8.

About tmWakeUp

The following sentence, also on page 3-8 in *Inside Macintosh: Processes*, is incorrect: "The tmWakeUp field contains the time at which the Time Manager task specified by tmAddr was last executed (or 0 if it has not yet been executed)." It should say: "The `tmWakeUp` field contains the time at which the Time Manager task specified by `tmAddr` is scheduled to be executed (or 0 if it has not yet been executed)."

Warning: Since the format of that field is undocumented and used internally by the Time Manager, developers are strongly discouraged anyway from performing any kind of calculation or comparison on the value of this field, since that format could change in the future.

Another Way to Perform Time Measurement

Another way to perform time measurement would be to use the Microseconds call, which is much easier to use and less likely to change in the future:

```
pascal void Microseconds(UnsignedWide *microseconds);
```

Warning: Currently, even with the most recent system software (7.5.3 revision 2) on a PCI Power Macintosh, both the Time Manager calls and the Microseconds

call are still in 68K code and thus are executed by the emulator. If you call them from PowerPC code, you'll get a switch from PowerPC code to the 68K emulator, so the values returned are incorrect by a few 10s of microseconds. This means that you have to be careful when using either of them to do time measurement. If you do use Microseconds, then your time measurement is done by the difference of the 2 values returned by Microseconds before and after the code you measure, and since the latency induced by the switches is the same in both case, then your time measurement is correct. If you do use the Time Manager way of performing time measurement, however, the `tmCount` field may be off by a few 10s of microsecond.

Summary

The following points explain what you should and should not do in working with the Time Manager:

- Always set `tmReserved` to 0 before calling `InsXTime`.

- Set `tmWakeUp` to 0 before the first call to `InsXTime`, never look at it or modify it (except to set it to 0 in some cases, no other value is acceptable) afterwards.

- `tmCount` is only valid after a call to `RmvTime`.

- Microseconds might be a good alternate way.

Further Reference

- *Inside Macintosh: Processes*, Chapter 3, The Time Manager

- Denis G. Pelli's Web page at <http://rajsky.psych.nyu.edu/VideoToolbox/>

Acknowledgments

Thanks to Brian Bechtel, Drew Colace, Denis G. Pelli and Bob Wambaugh.