Time Manager Reference

Carbon > Resource Management

2006-03-08

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Time Manager Reference

Framework: CoreServices/CoreServices.h

Declared in Timer.h

Overview

The Time Manager allows applications and other software to schedule routines for execution at a later time. By suitably defining the routine that is to be executed later, you can use the Time Manager to accomplish a wide range of time-related activities. For example, because a routine can reschedule itself for later execution, the Time Manager allows your application to perform periodic or repeated actions. You can use the Time Manager to schedule routines for execution after a specified delay; set up tasks that run periodically; compute the time a routine takes to run; and coordinate and synchronize actions in the Macintosh computer.

The Time Manager provides a hardware-independent method of performing these time-related tasks. In general, you should use the Time Manager instead of timing loops, which can vary in duration because they depend on clock speed and interrupt-handling speed.

Carbon supports the Time Manager. However, the interface for callbacks will change because the current task record is accessible only from 68K code.

Functions by Task

Installing and Removing Tasks

`InstallTimeTask` (page 13) Deprecated in Mac OS X v10.4

Installs a task structure into the Time Manager task queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`InstallXTimeTask` (page 14) Deprecated in Mac OS X v10.4

Installs a task, taking advantage of the drift-free, fixed-frequency timing services of the extended Time Manager. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`RemoveTimeTask` (page 17) Deprecated in Mac OS X v10.4

Removes a task from the Time Manager queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Activating Tasks

`PrimeTimeTask` (page 16) Deprecated in Mac OS X v10.4
: Activates a task in the Time Manager queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Measuring Time

`Microseconds` (page 8)
: Determines the number of microseconds that have elapsed since system startup time.

Working With Your Time Manager Callback Function

`NewTimerUPP` (page 8)
: Creates a new universal procedure pointer (UPP) to your Time Manager task callback. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`InvokeTimerUPP` (page 7)
: Invokes your Time Manager task callback function. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`DisposeTimerUPP` (page 7)
: Disposes of the universal procedure pointer (UPP) to your Time Manager task callback function. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Obsolete Functions

`InsTime` (page 14) Deprecated in Mac OS X v10.4
: Installs a task record into the Time Manager task queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`InsXTime` (page 15) Deprecated in Mac OS X v10.4
: Installs an extended task record into the Time Manager task queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`PrimeTime` (page 15) Deprecated in Mac OS X v10.4
: Activates a task in the Time Manager queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

`RmvTime` (page 18) Deprecated in Mac OS X v10.4
: Remove a task from the Time Manager queue. (Deprecated. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Functions

DisposeTimerUPP

Disposes of the universal procedure pointer (UPP) to your Time Manager task callback function. (**Deprecated.** Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
void DisposeTimerUPP (
   TimerUPP userUPP
);
```

Parameters

userUPP

A UPP to your callback function.

Discussion

See the callback `TimerProcPtr` (page 9) for more information.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Declared In

`Timer.h`

InvokeTimerUPP

Invokes your Time Manager task callback function. (**Deprecated.** Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
void InvokeTimerUPP (
   TMTaskPtr tmTaskPtr,
   TimerUPP userUPP
);
```

Parameters

tmTaskPtr

A pointer to a structure of type `TMTask` containing the information about the task.

userUPP

A UPP to your callback function.

Discussion

See the callback `TimerProcPtr` (page 9) for more information.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Timer.h`

Microseconds

Determines the number of microseconds that have elapsed since system startup time.

```
void Microseconds (
   UnsignedWide *microTickCount
);
```

Parameters

microTickCount

The number of microseconds elapsed since system startup.

Availability
Available in CarbonLib 1.0 and later when running Mac OS 8.1 or later.
Available in Mac OS X 10.0 and later.

Declared In
`Timer.h`

NewTimerUPP

Creates a new universal procedure pointer (UPP) to your Time Manager task callback. (**Deprecated.** Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
TimerUPP NewTimerUPP (
   TimerProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your Time Manager event callback function. For information on how to create a Time Manager event callback see `TimerProcPtr` (page 9)

Return Value
A UPP to your callback function. See the description of the `TimerUPP` data type.

Discussion
See the callback `TimerProcPtr` (page 9) for more information.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.

Declared In
`Timer.h`

Callbacks

TimerProcPtr

Defines a pointer to your application-defined Time Manager task that is executed after a specified delay.

```
typedef void (*TimerProcPtr) (
    TMTaskPtr tmTaskPtr
);
```

If you name your function `MyTimerProc`, you would declare it like this:

```
void MyTimerProc (
    TMTaskPtr tmTaskPtr
);
```

Parameters

`tmTaskPtr`
A pointer to a structure of type `TMTask` containing the information about the task.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Timer.h`

Data Types

TimerUPP

Defines a data type for the `TimerProcPtr` callback function.

```
typedef TimerProcPtr TimerUPP;
```

Discussion
For more information, see the description of the `TimerProcPtr` (page 9) callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Timer.h`

TMTask

Contains information for a Time Manager task.

```
struct TMTask {
    QElemPtr qLink;
    short qType;
    TimerUPP tmAddr;
    long tmCount;
    long tmWakeUp;
    long tmReserved;
};
typedef struct TMTask TMTask;
typedef TMTask * TMTaskPtr;
```

Fields

`qLink`
: A pointer to the next element in the Time Manager queue. This field is used internally by the Time Manager.

`qType`
: The type of queue. The Time Manager automatically sets this field to the appropriate value. The high-order bit of this field is a flag that indicates whether the task is active.

`tmAddr`
: A pointer to the function that is to execute after the delay specified in a call to `PrimeTime`.

`tmCount`
: Reserved in the original Time Manager. In the revised or extended Time Manager, the amount of time remaining until the task's scheduled execution time. This field is valid only after you call `RmvTime` with a task that has not yet executed.

`tmWakeUp`
: In the extended Time Manager, the time when the task specified in the `tmAddr` field was last executed. This field is used internally by the Time Manager. You should set it to 0 when you first install a task structure.

`tmReserved`
: Reserved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`Timer.h`

Constants

Active Task Constant

Defines a constant for an active task.

```
enum {
    kTMTaskActive = (1L << 15)
};
```

Constants

`kTMTaskActive`

The high bit of the `qType` field in the `TMTask` structure is set if the task is active.

Available in Mac OS X v10.0 and later.

Declared in `Timer.h`.

Result Codes

The most common result codes returned by Time Manager is `noErr`, which has a value of 0.

Gestalt Constants

You can check for version and feature availability information by using the Time Manager Version selectors defined in the Gestalt Manager. For more information see *Inside Mac OS X: Gestalt Manager Reference*.

Deprecated Time Manager Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

InstallTimeTask

Installs a task structure into the Time Manager task queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
OSErr InstallTimeTask (
   QElemPtr tmTaskPtr
);
```

Parameters

tmTaskPtr

A pointer to an original task structure to be installed in the queue.

Return Value

A result code. See "Time Manager Result Codes" (page 11).

Discussion

The `InstallTimeTask` function adds the Time Manager task structure specified by the `tmTaskPtr` parameter to the Time Manager queue. Your application should fill in the `tmAddr` field of the task structure and should set the `tmCount` field to 0. The `tmTaskPtr` parameter must point to an original Time Manager task structure.

With the revised and extended Time Managers, you can set the `tmAddr` field to `NULL` if you do not want a task to execute when the delay passed to the `PrimeTime` function expires. Also, the revised Time Manager resets the high-order bit of the `qType` field to 0 when you call the `InsTime` function.

The `InstallTimeTask` function, which returns a value of type `OSErr`, takes the place of `InsTime`.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability

Available in CarbonLib 1.0.2 and later when running Mac OS 9.1 or later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In
`Timer.h`

InstallXTimeTask

Installs a task, taking advantage of the drift-free, fixed-frequency timing services of the extended Time Manager. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
OSErr InstallXTimeTask (
   QElemPtr tmTaskPtr
);
```

Parameters

tmTaskPtr

A pointer to an extended task structure to be installed in the queue.

Return Value
A result code. See "Time Manager Result Codes" (page 11).

Discussion
The `InstallXTimeTask` function adds the Time Manager task structure specified by `tmTaskPtr` to the Time Manager queue. Use `InstallXTimeTask` only if you wish to use the drift-free, fixed-frequency timing services of the extended Time Manager; use `InstallTimeTask` in all other cases. The `tmTaskPtr` parameter must point to an extended Time Manager task structure. Your application must fill in the `tmAddr` field of that task. You should set the `tmWakeUp` and `tmReserved` fields to 0 the first time you call `InsXTime`.

With the extended Time Manager, you can set `tmAddr` to `NULL` if you do not want a task to execute when the delay passed to `PrimeTime` expires. Also, `InsXTime` resets the high-order bit of the `qType` field to 0.

The `InstallXTimeTask` function, which returns a value of type `OSErr`, takes the place of InsXTime.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0.2 and later when running Mac OS 9.1 or later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`Timer.h`

InsTime

Installs a task record into the Time Manager task queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Not Recommended

```
void InsTime (
   QElemPtr tmTaskPtr
);
```

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0 and later when running Mac OS 8.1 or later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`Timer.h`

InsXTime

Installs an extended task record into the Time Manager task queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Not Recommended

```
void InsXTime (
   QElemPtr tmTaskPtr
);
```

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0 and later when running Mac OS 8.1 or later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`Timer.h`

PrimeTime

Activates a task in the Time Manager queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Not Recommended

```
void PrimeTime (
   QElemPtr tmTaskPtr,
   long count
);
```

Discussion
This function is deprecated. You should use the function `PrimeTimeTask` (page 16) instead.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability
Available in CarbonLib 1.0 and later when running Mac OS 8.1 or later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`Timer.h`

PrimeTimeTask

Activates a task in the Time Manager queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
OSErr PrimeTimeTask (
   QElemPtr tmTaskPtr,
   long count
);
```

Parameters

tmTaskPtr
A pointer to a task structure already installed in the queue.

count
The desired delay before execution of the task.

Return Value
A result code. See "Time Manager Result Codes" (page 11).

Discussion
The `PrimeTimeTask` function schedules the task specified by the `tmAddr` field of the structure pointed to by the *tmTaskPtr* parameter for execution after the delay specified by the *count* parameter has elapsed.

If the *count* parameter is a positive value, it is interpreted as milliseconds. If *count* is a negative value, it is interpreted in negated microseconds. Microsecond delays are allowable only in the revised and extended Time Managers.

The task record specified by the *tmTaskPtr* parameter must already be installed in the queue (by a previous call to the functions InstallTimeTask (page 13) or InstallXTimeTask (page 14)) before your application calls the PrimeTimeTask function. The PrimeTimeTask function returns immediately, and the specified task is executed after the specified delay has elapsed. If you call the PrimeTimeTask function with a time delay of 0, the task runs as soon as interrupts are enabled.

In the revised and extended Time Managers, the PrimeTimeTask function sets the high-order bit of the qType field to 1. In addition, any value of the *count* parameter that exceeds the maximum millisecond delay is reduced to the maximum. If you stop an unexpired task (by calling the function RemoveTimeTask (page 17)) and then reinstall it (by calling the InstallXTimeTask function), you can continue the previous delay by calling the PrimeTimeTask function with the *count* parameter set to 0.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability

Available in CarbonLib 1.0.2 and later when running Mac OS 9.1 or later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

Timer.h

RemoveTimeTask

Removes a task from the Time Manager queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

```
OSErr RemoveTimeTask (
   QElemPtr tmTaskPtr
);
```

Parameters

tmTaskPtr

A pointer to a task structure to be removed from the queue.

Return Value

A result code. See "Time Manager Result Codes" (page 11).

Discussion

The RemoveTimeTask function removes the Time Manager task structure specified by the *tmTaskPtr* parameter from the Time Manager queue. In both the revised and extended Time Managers, if the specified task record is active (that is, if it has been activated but the specified time has not yet elapsed), the tmCount field of the task structure returns the amount of time remaining. To provide the greatest accuracy, the unused time is reported as negated microseconds if that value is small enough to fit into the tmCount field (even if the delay was originally specified in milliseconds); otherwise, the unused time is reported in positive milliseconds. If the time has already expired, the tmCount field contains 0.

In the revised and extended Time Managers, the RemoveTimeTask function sets the high-order bit of the qType field to 0.

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability

Available in CarbonLib 1.0.2 and later when running Mac OS 9.1 or later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`Timer.h`

RmvTime

Remove a task from the Time Manager queue. (Deprecated in Mac OS X v10.4. Use Carbon Event Loop timers or Cocoa NSTImers instead.)

Not Recommended

```
void RmvTime (
   QElemPtr tmTaskPtr
);
```

Special Considerations

Carbon Event timers and Cocoa NSTimers provide a simpler and more efficient way to handle timed or periodic tasks. For more information about using Carbon Event timers, see the timers section in *Carbon Event Manager Programming Guide*. For information about NSTimers, see *Timer Programming Topics for Cocoa*. Both Carbon event timers and NSTimers are built on top of the lower-level Core Foundation CFRunLoop timers. For CFRunLoop information, see *Run Loops*.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.1 or later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`Timer.h`

Document Revision History

This table describes the changes to *Time Manager Reference*.

Date	Notes
2006-03-08	Added information about deprecated functions.
2003-04-01	Added abstracts for deprecated functions.
2003-02-01	Updated formatting.
	Removed reference to time.h and all functions supposedly associated with it. Man pages are available for the functions `asctime`, `clock`, `ctime`, `difftime`, `gmtime`, `localtime`, `mktime`, `strftime`, and `time`.
	Documented and grouped functions in the old Miscellaneous section.
2001-07-01	Last version of this document.

Document Revision History

Index