

This chapter discusses clock components. **Clock components** provide timing information. In QuickTime, the Movie Toolbox is the primary client of clock components. Applications seldom call clock components directly. However, you may want to develop your own clock component for use by the Movie Toolbox. Therefore, this chapter focuses on what you must do to create a clock component.

- “About Clock Components” presents some general information about clock components.
- “Clock Components Reference” describes the constants, data structures, and functions that are specific to clock components.
- “Summary of Clock Components” provides summaries of the clock component constants, data structures, and functions in C and in Pascal.

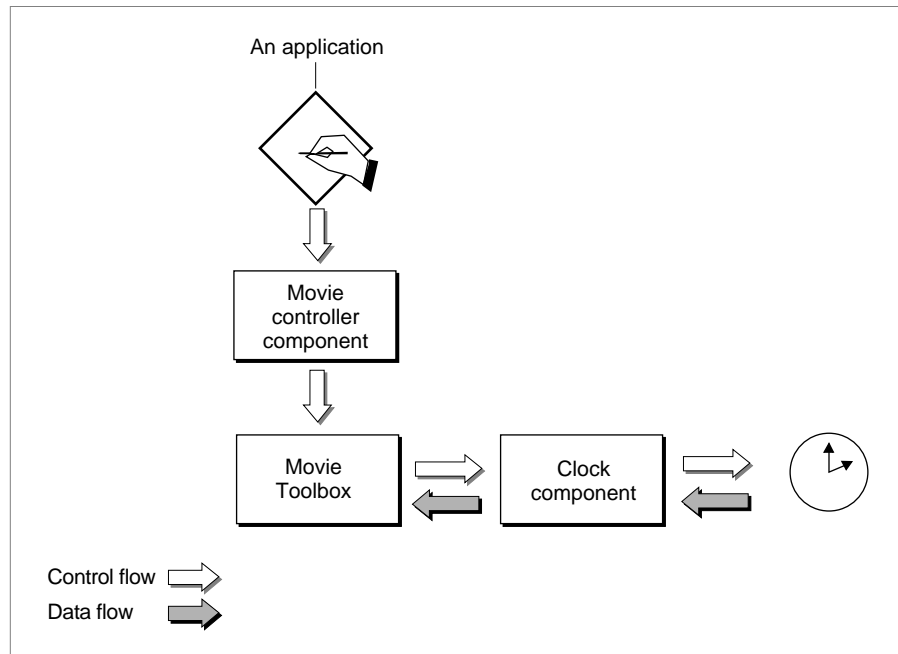
Before learning about clock components, you must be familiar with QuickTime time bases. See the chapter “Movie Toolbox” in *Inside Macintosh: QuickTime* for a complete description of time bases and of the Movie Toolbox functions that support time bases.

## About Clock Components

---

Clock components provide two basic services: they generate time information and schedule time-based callback events. In QuickTime, the Movie Toolbox is the primary user of clock components. Specifically, the Movie Toolbox uses clock components to provide basic timing to time bases. In general, clock components derive their timing information from some external source. For example, a clock component could use the Macintosh tick count to provide its basic timing. Alternatively, a clock component could use some special hardware installed in the Macintosh computer to provide its basic timing. Figure 11-1 shows the relationships between an application, the movie controller component, the Movie Toolbox, and a clock component.

**Figure 11-1** Relationships of an application, the movie controller component, the Movie Toolbox, and a clock component



Clock components may also support time-based callback events. The Movie Toolbox's time base functions allow applications and other programs to schedule functions to be called in specified circumstances. Since time bases derive their time information from clock components, ultimate responsibility for servicing these callback functions also falls to clock components. The Movie Toolbox provides a set of support functions that your clock component can use to manage its callback events—these functions are described later in this chapter.

Your clock component is not required to support callback functions. You can delegate this responsibility to another clock component. “Component Capability Flags for Clocks” on page 11-5 describes how you can tell the Component Manager that your clock component does not support callback functions.

## Clock Components Reference

---

This section describes the constants, data type, and functions that are specific to clock components.

### Component Capability Flags for Clocks

---

The Component Manager allows you to specify information about your component's capabilities in the `componentFlags` field of the component description structure. Apple has defined two component flags for clock components. These flags specify information about the capabilities of the clock component. You set these flags in the `componentFlags` field of your component's component description structure. You can use the following constants to manipulate these flags. You should set them appropriately for your clock. For more on the component description structure, see the chapter "Component Manager" in *Inside Macintosh: More Macintosh Toolbox*.

```
enum {
    kClockRateIsLinear = 1,           /* clock keeps constant
                                      rate */
    kClockImplementsCallbacks = 2    /* clock supports callback
                                      events */
};
```

`kClockRateIsLinear`

Indicates that your clock maintains a constant rate. Most clocks that you deal with in the everyday world fall into this category. An example of a clock with an irregular rate is a clock that is dependent on the position of the Macintosh computer's mouse—the clock's rate might change depending upon where the user moves the mouse. Set this flag to 1 if your clock has a constant rate.

`kClockImplementsCallbacks`

Indicates that your clock supports callback events. Set this flag to 1 if your clock supports callback events.

You should set the `componentFlags` field appropriately in the component description structure that is associated with your clock component.

## Component Types for Clocks

---

Apple has defined a type value and a number of subtype values for clock components. All clock components have a component type value of 'clock'. The component subtype value indicates the type of clock. You can use the following constants to specify these type and subtype values.

```
#define clockComponentType      'clock' /* clock component type */
#define systemTickClock        'tick'  /* system tick clock */
#define systemSecondClock      'seco'  /* system seconds clock */
#define systemMillisecondClock 'mill'  /* system millisecond clock */
#define systemMicrosecondClock 'micr'  /* system microsecond clock */
```

## Data Type

---

The clock component data structure is a private data structure. Programs that use your clock component never change the contents of this data structure directly. Your clock component provides functions that allow programs to use this data structure.

The callback header structure specifies the callback function for an operation. Your application can obtain callback function identifiers by calling its clock component's `ClockNewCallBack` function (described on page 11-10).

The `QTCallBackHeader` data type defines the callback header structure.

```
struct QTCallBackHeader {
    long      callBackFlags; /* flags used by clock
                             component to communicate
                             scheduling data about
                             callback to Movie Toolbox */
    long      reserved1;     /* reserved for use by Apple */
    char      qtPrivate[40]; /* reserved for use by Apple */
};
```

### Field descriptions

#### callBackFlags

Contains flags that your component can use to communicate scheduling information about the callback event to the Movie Toolbox. This scheduling information tells the Movie Toolbox what time base events your clock component needs to know about in order to support the callback event. The following flags are defined (all other flags must be set to 0):

```
enum {
    qtcBNeedsRateChanges = 1, /* clock needs to
                               know about rate
                               changes */
```

```

    qtcNeedsTimeChanges = 2    /* clock needs to
                                know about time
                                changes */

    qtcNeedsStartStopChanges
                                = 4    /* clock needs to
                                know about time
                                base changes */

};

```

`qtcNeedsRateChanges`

Indicates that your clock component needs to know about rate changes. If you set this flag to 1, the Movie Toolbox calls your `ClockRateChanged` function (described on page 11-16) whenever the rate of the callback event's time base changes.

`qtcNeedsTimeChanges`

Indicates that your clock component needs to know about time changes. If you set this flag to 1, the Movie Toolbox calls your `ClockTimeChanged` function (described on page 11-15) whenever a program changes the time value of the time base, or when the time value changes by an amount that is different from the time base's rate.

`qtcNeedsStartStopChanges`

Indicates that your clock component needs to know about the time base's start and stop changes. If you set this flag to 1, the Movie Toolbox calls your `ClockStartStopChanged` function (described on page 11-16) whenever a program changes the start or stop time of the time base.

`reserved1`    Reserved for use by Apple.

`qtPrivate`    Reserved for use by Apple.

## Clock Component Functions

This section describes the functions that are provided by clock components. These functions are described from the perspective of the Movie Toolbox, the entity that is most likely to call clock components. If you are developing a clock component, your component must behave as described here.

## Clock Components

This section has been divided into the following topics:

- “Getting the Current Time” describes the function that allows the Movie Toolbox to obtain the current time from a clock component.
- “Using the Callback Functions” discusses the functions that allow clock components to help applications define and schedule time base callback functions.
- “Managing the Time” describes functions that help clock components manage their time correctly.

If you are developing an application that uses clock components, you should read the next section, “Getting the Current Time.”

If you are developing a clock component, you need to be familiar with all the functions described in this section.

**Note**

Your application can call any clock component function at interrupt time, except for the `ClockNewCallBack` and `ClockDisposeCallBack` functions (described on page 11-10 and page 11-14, respectively). In addition, your application should not call the Component Manager’s `OpenComponent` and `CloseComponent` functions at interrupt time. ♦

You can use the following constants to refer to the request codes for each of the functions that your clock component must support:

```
/* constants to refer to request codes for supported functions */
enum {
    kClockGetTimeSelect          = 0x1, /* ClockGetTime */
    kClockNewCallBackSelect      = 0x2, /* ClockNewCallBack */
    kClockDisposeCallBackSelect  = 0x3, /* ClockDisposeCallBack */
    kClockCallMeWhenSelect       = 0x4, /* ClockCallMeWhen */
    kClockCancelCallBackSelect   = 0x5, /* ClockCancelCallBack */
    kClockRateChangedSelect      = 0x6, /* ClockRateChanged */
    kClockTimeChangedSelect      = 0x7, /* ClockTimeChanged */
    kClockSetTimeBaseSelect      = 0x8, /* ClockSetTimeBase */
    kClockStartStopChangedSelect = 0x9, /* ClockStartStopChanged */
    kClockGetRateSelect          = 0xA, /* ClockGetRate */
};
```

## Getting the Current Time

---

Clock components provide a single function that allows the Movie Toolbox to obtain the current time.

## ClockGetTime

---

The `ClockGetTime` function allows the Movie Toolbox to obtain the current time according to the specified clock.

```
pascal ComponentResult ClockGetTime (ComponentInstance aClock,
                                     TimeRecord *out);
```

**aClock** Specifies the clock for the operation. You obtain this identifier from the Component Manager's `OpenComponent` function. See the chapter "Component Manager" in *Inside Macintosh: More Macintosh Toolbox* for details.

**out** Contains a pointer to a time structure. (For details on the time structure, see the chapter "Movie Toolbox" in *Inside Macintosh: QuickTime*.) The clock component updates this structure with the current time information. Specifically, the clock component sets the `value` and `scale` fields in the time structure. Your clock component should always return values in its native time scale—this time scale does not change during the life of the component connection.

### DESCRIPTION

The `ClockGetTime` function is the most important function for most clock components. The Movie Toolbox calls this function very often, so it should be fast.

## Using the Callback Functions

---

Applications that use QuickTime time bases may define callback functions that are associated with a specific time base. Applications can then use these callback functions to perform activities that are triggered by temporal events, such as a certain time being reached or a specified rate being achieved. The time base functions of the Movie Toolbox interact with clock components to schedule the invocation of these callback functions—your clock component is responsible for calling the callback function at its scheduled time.

## Clock Components

The functions described in this section are called by the Movie Toolbox to support applications that define time base callback functions. For more information about time base callback functions, see the chapter “Movie Toolbox” in *Inside Macintosh: QuickTime*. Note that your clock component can delegate its callback events to another component by calling the Component Manager’s `DelegateComponent` function, which is described in the chapter “Component Manager” in *Inside Macintosh: More Macintosh Toolbox*.

The `ClockNewCallBack` function allows your clock component to allocate the memory to support a new callback event. When an application discards a callback event, the Movie Toolbox calls your clock component’s `ClockDisposeCallBack` function.

The Movie Toolbox calls your clock component’s `ClockCallMeWhen` function when an application wants to schedule a callback event. When the callback function is to be invoked to service the event, the Movie Toolbox calls your component’s `ClockCancelCallBack` function so that you can remove the callback event from the list of scheduled events.

## ClockNewCallBack

---

Your component’s `ClockNewCallBack` function allocates the memory for a new callback event. The Movie Toolbox calls this function when an application defines a time base callback event with the Movie Toolbox’s `NewCallBack` function. The callback event created at this time is not active until it has been scheduled. An application schedules a callback event by calling the Movie Toolbox’s `CallMeWhen` function.

Your component allocates the memory required to support the callback event. The memory must be in a locked block and must begin with a callback header structure. This structure is described in “Data Type,” which begins on page 11-6.

You should not call this function at interrupt time.

```
pascal QTCallBack ClockNewCallBack (ComponentInstance aClock,
                                   TimeBase tb,
                                   short callBackType);
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Applications obtain this identifier from the Component Manager’s <code>OpenComponent</code> function.   |
| <code>tb</code>     | Specifies the callback event’s time base. Typically, your component does not need to save this specification. You can use the Movie Toolbox’s <code>GetCallBackTimeBase</code> function to determine the callback event’s time base when it is invoked (see the discussion of time bases in the chapter “Movie Toolbox” in <i>Inside Macintosh: QuickTime</i> for more information about this function). |



## Clock Components

`callbackType`

Specifies when the callback event is to be invoked. The value of this field governs how your component interprets the data supplied in the `param1`, `param2`, and `param3` parameters to the `ClockCallMeWhen` function, which is described in the next section. The following three values are valid for this parameter:

`callbackAtTime`

Indicates that the callback event is to be invoked at a specified time. The Movie Toolbox supplies this time to your component in the parameter data of the `ClockCallMeWhen` function (described in the next section).

`callbackAtRate`

Indicates that the callback event is to be invoked when the rate for the time base reaches a specified value. The Movie Toolbox supplies this value to your component in the parameter data of the `ClockCallMeWhen` function.

`callbackAtTimeJump`

Indicates that the callback event is to be invoked when a program changes the time value for the time base.

In addition, if the high-order bit of the `callbackType` parameter is set to 1 (this bit is defined by the `callbackAtInterrupt` flag), the callback event may be invoked at interrupt time.

**DESCRIPTION**

Your clock component allocates the memory for the event and returns a pointer to that memory. If your clock component cannot satisfy the request or detects invalid or unsupported parameter values, you should set the `QTCallback` result to `nil`.

Your component can allocate an arbitrarily large piece of memory for the callback event. That memory must begin with a callback header structure, which must be initialized to 0.

**ClockCallMeWhen**

Your clock component's `ClockCallMeWhen` function schedules a callback event for invocation. The Movie Toolbox calls this function when an application schedules a callback event using the `CallMeWhen` function of the Movie Toolbox (described in the chapter "Movie Toolbox" in *Inside Macintosh: QuickTime*).

## Clock Components

The Movie Toolbox passes the parameter data from its `CallMeWhen` function to your component in the `param1`, `param2`, and `param3` parameters to this function. Your clock component interprets these parameters based on the value of the `callBackType` parameter to the `ClockNewCallBack` function (see page 11-10).

```
pascal ComponentResult ClockCallMeWhen (ComponentInstance aClock,
                                         QTCallBack cb,
                                         long param1,
                                         long param2,
                                         long param3);
```

**aClock** Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

**cb** Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` function.

**param1** Contains data supplied to the Movie Toolbox in the `param1` parameter to the `CallMeWhen` function. Your component interprets this parameter based on the value of the `callBackType` parameter to the `ClockNewCallBack` function.

If `callBackType` is set to `callBackAtTime`, `param1` contains QuickTime callback flags indicating when to invoke the callback function. The following values are defined:

`triggerTimeFwd`

Indicates that the callback function should be called at the time specified by `param2` only when time is moving forward (positive rate). The value of this flag is 0x0001.

`triggerTimeBwd`

Indicates that the callback function should be called at the time specified by `param2` only when time is moving backward (negative rate). The value of this flag is 0x0002.

`triggerTimeEither`

Indicates that the callback function should be called at the time specified by `param2` without regard to direction. The value of this flag is 0x0003.

If `callBackType` is set to `callBackAtRate`, `param1` contains flags indicating when to invoke the callback function.

The following values are defined:

`triggerRateChange`

Indicates that the callback function should be called whenever the rate changes. The value of this flag is 0.

`triggerRateLT`

Indicates that the callback function should be called when the rate changes to a value less than that specified by `param2`. The value of this flag is 0x0004.

|                     |  |
|---------------------|--|
| triggerRateGT       | Indicates that the callback function should be called when the rate changes to a value greater than that specified by param2. The value of this flag is 0x0008.  |
| triggerRateEqual    | Indicates that the callback function should be called when the rate changes to a value equal to that specified by param2. The value of this flag is 0x0010.  |
| triggerRateLTE      | Indicates that the callback function should be called when the rate changes to a value that is less than or equal to that specified by param2. The value of this flag is 0x0014.   |
| triggerRateGTE      | Indicates that the callback function should be called when the rate changes to a value that is less than or equal to that specified by param2. The value of this flag is 0x0018.   |
| triggerRateNotEqual | Indicates that the callback function should be called when the rate changes to a value that is not equal to that specified by param2. The value of this flag is 0x001C.  |
| param2              | <p>Contains data supplied to the Movie Toolbox in the param2 parameter to the CallMeWhen function (see page 11-11). Your component interprets this parameter based on the value of the callbackType parameter to the ClockNewCallBack function, described on page 11-10.</p> <p>If callbackType is set to callbackAtTime, param2 contains the time value at which your component should invoke the callback function for this event. The param1 parameter contains flags affecting when you should call the function.</p> <p>If callbackType is set to callbackAtRate, param2 contains the rate value at which your component should invoke the callback function for this event. The param1 parameter contains flags affecting when you should call the function.</p> |
| param3              | Contains data supplied to the Movie Toolbox in the param3 parameter to the CallMeWhen function. If qtType is set to callbackAtTime, param3 contains the time scale in which to interpret the time value that is stored in param2.  |

**DESCRIPTION**

The Movie Toolbox maintains control information about the callback event. Your clock component only needs to maintain the invocation schedule. For example, the Movie Toolbox saves the address of the callback event, its reference constant, and the value of the A5 register. In addition, the Movie Toolbox prevents applications from scheduling a single callback event more than once.

## Clock Components

If your clock component successfully schedules the callback event, you should call the `AddCallbackToTimeBase` function (described on page 11-18) to add it to the list of callback events for the corresponding time base. If your component cannot schedule the callback event, it should return an appropriate error.

## ClockCancelCallback

---

Your clock component's `ClockCancelCallback` function removes the specified callback event from the list of scheduled callback events for a time base.

```
pascal ComponentResult ClockCancelCallback
                                (ComponentInstance aClock,
                                QTCallback cb)
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Your application obtains this identifier from the Component Manager's <code>OpenComponent</code> function.                            |
| <code>cb</code>     | Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's <code>ClockNewCallback</code> function (described on page 11-10). |

### DESCRIPTION

The Movie Toolbox calls this function when an application cancels its callback event by calling `CancelCallback`. The Movie Toolbox also calls this function whenever it executes the callback event, thus removing it from the list of scheduled callback events. The application is then responsible for rescheduling the event, if appropriate.

If your clock component successfully cancels the callback event, you should call the `RemoveCallbackFromTimeBase` function, described on page 11-19, so that the Movie Toolbox can remove the callback event from its list of scheduled events.

## ClockDisposeCallback

---

Your clock component's `ClockDisposeCallback` function disposes of the memory associated with the specified callback event.

```
pascal ComponentResult ClockDisposeCallback
                                (ComponentInstance aClock,
                                QTCallback cb);
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's <code>OpenComponent</code> function. |
|---------------------|--|

**cb** Specifies the callback event for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` function (described on page 11-10).

**DESCRIPTION**

The Movie Toolbox calls this function when an application discards its callback event by calling the `DisposeCallBack` function. Your clock component should cancel the callback event before you dispose of it.

You should not call this function at interrupt time.

**Managing the Time**

Clock components provide several functions that allow the Movie Toolbox to alert your component to changes in its environment. Three of these functions, `ClockTimeChanged`, `ClockRateChanged`, and `ClockStartStopChanged`, are associated with application callback functions and help your component determine whether to invoke the callback function. The fourth, the `ClockSetTimeBase` function, tells your clock component about the time base it is supporting.

**ClockTimeChanged**

The Movie Toolbox calls your component's `ClockTimeChanged` function whenever the callback's time base time value is set. The Movie Toolbox calls this function only if the `qtcBNeedsTimeChanges` flag is set to 1 in the `callBackFlags` field of the QuickTime callback header structure allocated by your clock component (see "Data Type" beginning on page 11-6 for more information).

```
pascal ComponentResult ClockTimeChanged
                                (ComponentInstance aClock,
                                QTCallBack cb);
```

**aClock** Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's `OpenComponent` function.

**cb** Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's `ClockNewCallBack` function.

**DESCRIPTION**

The Movie Toolbox calls this function once for each qualified callback function associated with the time base. Note that the Movie Toolbox calls this function only for callback events that are currently scheduled.

## ClockRateChanged

---

The Movie Toolbox calls your component's `ClockRateChanged` function whenever the callback's time base rate changes. The Movie Toolbox calls this function only if the `qtcBNeedsRateChanges` flag is set to 1 in the `callbackFlags` field of the callback header structure in the `QTCallbackHeader` structure allocated by your clock component (see "Data Type" beginning on page 11-6 for more information about the callback header structure).

```
pascal ComponentResult ClockRateChanged (ComponentInstance aClock,
                                         QTCallback cb);
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's <code>OpenComponent</code> function.                           |
| <code>cb</code>     | Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's <code>ClockNewCallback</code> function (described on page 11-10). |

### DESCRIPTION

The Movie Toolbox calls this function once for each qualified callback function associated with the time base. Note that the Movie Toolbox calls this function only for callback events that are currently scheduled.

## ClockStartStopChanged

---

The Movie Toolbox calls your component's `ClockStartStopChanged` function whenever the start or stop time of the callback's time base changes. The Movie Toolbox calls this function only if the `qtcBNeedsStartStop` flag is set to 1 in the `callbackFlags` field of the callback header structure in the `QTCallbackHeader` structure allocated by your clock component (see "Data Type" beginning on page 11-6 for more information about the callback header structure).

```
pascal ComponentResult ClockStartStopChanged
    (ComponentInstance aClock, QTCallback cb,
     Boolean startChanged,
     Boolean stopChanged);
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's <code>OpenComponent</code> function. |
|---------------------|--|

## Clock Components

|                           |  |
|---------------------------|--|
| <code>cb</code>           | Specifies the callback for the operation. The Movie Toolbox obtains this value from your component's <code>ClockNewCallBack</code> function (described on page 11-10). |
| <code>startChanged</code> | Indicates that the start time of the time base associated with the clock component instance has changed.   |
| <code>stopChanged</code>  | Indicates that the stop time of the time base associated with the clock component instance has changed.  |

**DESCRIPTION**

The Movie Toolbox calls this function once for each qualified callback function associated with the time base. Note that the Movie Toolbox calls this function only for callback events that are currently scheduled.

**ClockSetTimeBase**

The Movie Toolbox calls your component's `ClockSetTimeBase` function when an application creates a time base that uses your clock component. The `tb` parameter indicates the time base that is associated with your clock.

```
pascal ComponentResult ClockSetTimeBase (ComponentInstance aClock,
                                          TimeBase tb);
```

|                     |  |
|---------------------|--|
| <code>aClock</code> | Specifies the clock for the operation. Applications obtain this identifier from the Component Manager's <code>OpenComponent</code> function. |
| <code>tb</code>     | Specifies the time base that is associated with the clock.   |

**DESCRIPTION**

Your clock component may need to know its time base if the rate or time value of the time base can be changed without using Movie Toolbox functions. This could be the case if your clock supports an external clock. Under these circumstances, the Movie Toolbox cannot use the `ClockRateChanged` and `ClockTimeChanged` functions (described on page 11-16 and page 11-15, respectively) to alert your component to changes in its environment. Instead, your component can use the time base provided here to seed the `GetFirstCallBack` function, described on page 11-20, and then scan all its associated callback functions.

## Movie Toolbox Clock Support Functions

---

The Movie Toolbox provides a number of support functions for clock components. All of these functions help your component manage its associated callback functions. Your clock component may call any of these functions at interrupt time. These functions should only be called by clock components.

Use the `AddCallbackToTimeBase` function to add a callback event to the list of scheduled callback events maintained by the Movie Toolbox. You should use the `RemoveCallbackFromTimeBase` function to remove a callback event from the list.

When your clock component determines that it is time to invoke a callback function, you should use the `ExecuteCallback` function to cause the Movie Toolbox to call the function.

If your clock component needs to scan all its associated callback events, you can use the `GetFirstCallback` and `GetNextCallback` functions.

### AddCallbackToTimeBase

---

Your clock component uses the `AddCallbackToTimeBase` function to place a callback event into the list of scheduled callback events. The Movie Toolbox maintains this list.

```
pascal OSErr AddCallbackToTimeBase (QTCallback cb);
```

|                 |   |
|-----------------|---|
| <code>cb</code> | Specifies the callback event for the operation. Your clock component obtains this value from the parameters passed to your <code>ClockCallMeWhen</code> function (described on page 11-11). |
|-----------------|---|

#### DESCRIPTION

Your component should call the `AddCallbackToTimeBase` function when your `ClockCallMeWhen` function determines that your component can support the callback event (see “Using the Callback Functions,” which begins on page 11-9, for more information about the `ClockCallMeWhen` function).

If your component does not call this function, the Movie Toolbox does not notify your component of time, rate, or stop and start changes (via the `ClockRateChanged` and `ClockTimeChanged` functions, described on page 11-16 and page 11-15, respectively).



## ExecuteCallback

---

When your clock component determines that it is time to execute a callback function, your component should call the `ExecuteCallback` function.

```
pascal void ExecuteCallback (QTCallback cb);
```

**cb** Specifies the callback event for the operation. Your clock component obtains this value from the parameters passed to your `ClockCallMeWhen` function (described on page 11-11).

### DESCRIPTION

This function handles all the details of invoking the callback function properly. For example, the `ExecuteCallback` function queues the callback function correctly, according to the function's ability to execute at interrupt time (specified in the `callbackType` parameter to your `ClockNewCallback` function, described on page 11-10).

Before calling the application's function, the `ExecuteCallback` function cancels the callback event. In this manner, the callback event is prevented from executing twice in succession. It is up to the application, or the callback function itself, to reschedule the callback event.

### SPECIAL CONSIDERATIONS

This function sets the A5 register to the value it contained at the time the callback event was scheduled when calling the callback function.

Your clock component should not release the memory associated with the callback event at this time. You should do so only in your `ClockDisposeCallback` function (described on page 11-14). This is particularly important when a callback function cannot execute at interrupt time, since the Movie Toolbox schedules such functions for invocation at a later time.

## RemoveCallbackFromTimeBase

---

Your clock component uses the `RemoveCallbackFromTimeBase` function to remove a callback event from the list of scheduled callback events. The Movie Toolbox maintains this list.

```
pascal OSErr RemoveCallbackFromTimeBase (QTCallback cb);
```

**cb** Specifies the callback event for the operation. Your clock component obtains this value from the parameters passed to your `ClockCallMeWhen` function (described on page 11-11).

**DESCRIPTION**

Your component should call the `RemoveCallbackToTimeBase` function when your `ClockCancelCallback` function determines that your component can cancel the callback event (see “Using the Callback Functions” beginning on page 11-9 for more information about the `ClockCancelCallback` function).

**SPECIAL CONSIDERATIONS**

Your component should call the `RemoveCallbackFromTimeBase` function only for callback events that were successfully added to the schedule with the `AddCallbackToTimeBase` function (described on page 11-18).

**GetFirstCallback**

---

The `GetFirstCallback` function returns the first callback event associated with a specified time base. Your component can use this function, along with the `GetNextCallback` function (described in the next section), to scan all callback events associated with a time base.

```
pascal QTCallback GetFirstCallback (TimeBase tb);
```

**tb** Specifies the time base for the operation. Your component can obtain the time base reference from your `ClockSetTimeBase` function (described on page 11-17) or from the Movie Toolbox’s `GetCallbackTimeBase` function.

**DESCRIPTION**

The `GetFirstCallback` function returns the first callback event in the list managed for the specified time base. If there are no callback events associated with the time base, the `QTCallback` result is set to `nil`. Your component cannot assume that the Movie Toolbox maintains the callback list in any particular order.

## GetNextCallback

---

The `GetNextCallback` function returns the next callback event associated with a specified time base. Your component can use this function, along with the `GetFirstCallback` function (described in the previous section), to scan all callback events associated with a time base.

```
pascal QTCallback GetNextCallback (QTCallback cb);
```

**cb** Specifies the starting callback event for the operation. Your clock component obtains this value from the `GetFirstCallback` function or from previous calls to the `GetNextCallback` function.

### DESCRIPTION

The `GetNextCallback` function returns the next callback event in the list managed for the specified time base. If there are no more callback events associated with the time base, the returned QuickTime callback header structure is set to `nil`. Your component cannot assume that the Movie Toolbox maintains the callback list in any particular order.

## Summary of Clock Components

---

### C Summary

---

#### Constants

---

```

/* type value */
#define clockComponentType    'clock'    /* clock component */

/* subtype values */
#define systemTickClock      'tick'      /* system tick clock */
#define systemMicrosecondClock 'micr'    /* system microsecond clock */
#define systemSecondClock    'seco'      /* system second clock */
#define systemMillisecondClock 'mill'    /* system millisecond clock */

/* constants for manipulating clock component capability flags */
enum{
    kClockRateIsLinear = 1,              /* clock keeps constant rate */
    kClockImplementsCallbacks = 2        /* clock supports callback events */
};

#define ClockGetTime GetClockTime

/* constants to refer to request codes for supported functions */
enum {
    kClockGetTimeSelect      = 0x1,      /* ClockGetTime */
    kClockNewCallBackSelect   = 0x2,      /* ClockNewCallBack */
    kClockDisposeCallBackSelect = 0x3,    /* ClockDisposeCallBack */
    kClockCallMeWhenSelect    = 0x4,      /* ClockCallMeWhen */
    kClockCancelCallBackSelect = 0x5,      /* ClockCancelCallBack */
    kClockRateChangedSelect   = 0x6,      /* ClockRateChanged */
    kClockTimeChangedSelect    = 0x7,      /* ClockTimeChanged */
    kClockSetTimeBaseSelect    = 0x8,      /* ClockSetTimeBase */
    kClockStartStopChangedSelect = 0x9,    /* ClockStartStopChanged */
    kClockGetRateSelect        = 0xA,      /* ClockGetRate */
};

```

## Clock Components

```

enum {
    qtcBNeedsRateChanges      = 1, /* wants to know about rate changes */
    qtcBNeedsTimeChanges      = 2, /* wants to know about time changes */
    qtcBNeedsStartStopChanges = 4 /* wants to know when time base start
                                   or stop has changed */
};

/* values for callBackType parameter of ClockNewCallBack function that
   indicate when a callback event is to be invoked */

enum
{
    callBackAtTime      = 1, /* at specific time */
    callBackAtRate      = 2, /* when the rate for the time base
                               reaches a specific value */
    callBackAtTimeJump  = 3, /* when a program changes the time value
                               for a time base */
};

typedef unsigned short QTCallBackType;

/* callback equates--values for the parameter param1 of the
   ClockCallMeWhen function that indicate when the callback function should
   be called */

enum
{
    triggerTimeFwd      = 0x0001, /* when current time exceeds trigger time
                                     going forward */
    triggerTimeBwd      = 0x0002, /* when current time exceeds trigger time
                                     going backward */
    triggerTimeEither    = 0x0003, /* when curTime exceeds triggerTime going
                                     either direction */
    triggerRateLT        = 0x0004, /* when rate changes to less than trigger
                                     value */
    triggerRateGT        = 0x0008, /* when rate changes to greater than trigger
                                     value */
    triggerRateEqual     = 0x0010, /* when rate changes to equal trigger
                                     value */
    triggerRateLTE       = triggerRateLT | triggerRateEqual,
                               /* when rate changes to a value less than
                               or equal to param2 rate */
    triggerRateGTE       = triggerRateGT | triggerRateEqual,
                               /* when rate changes to value greater than
                               or equal to param2 rate */
};

```

## Clock Components

```

triggerRateNotEqual = triggerRateGT | triggerRateEqual | triggerRateLT,
                    /* when rate changes to value not equal to
                      param2 rate */
triggerRateChange   = 0,    /* whenever rate changes */
};
typedef unsigned short QTCallbackFlags;

```

## Data Type

---

```

struct QTCallbackHeader {
    long    callbackFlags;    /* flags used by clock component to
                              communicate scheduling data about
                              callback to Movie Toolbox */
    long    reserved1;       /* reserved for use by Apple */
    char    qtPrivate[40];   /* reserved for use by Apple */
};

```

## Clock Component Functions

---

### Getting the Current Time

```

pascal ComponentResult ClockGetTime
    (ComponentInstance aClock, TimeRecord *out);

```

### Using the Callback Functions

```

pascal QTCallback ClockNewCallback
    (ComponentInstance aClock, TimeBase tb,
     short callbackType);

pascal ComponentResult ClockCallMeWhen
    (ComponentInstance aClock, QTCallback cb,
     long param1, long param2, long param3);

pascal ComponentResult ClockCancelCallback
    (ComponentInstance aClock, QTCallback cb);

pascal ComponentResult ClockDisposeCallback
    (ComponentInstance aClock, QTCallback cb);

```

### Managing the Time

```

pascal ComponentResult ClockTimeChanged
    (ComponentInstance aClock, QTCallback cb);

pascal ComponentResult ClockRateChanged
    (ComponentInstance aClock, QTCallback cb);

```

```

pascal ComponentResult ClockStartStopChanged
    (ComponentInstance aClock, QTCallback cb,
     Boolean startChanged, Boolean stopChanged);
pascal ComponentResult ClockSetTimeBase
    (ComponentInstance clock, TimeBase tb);

```

### Movie Toolbox Clock Support Functions

---

```

pascal OSErr AddCallbackToTimeBase
    (QTCallback cb);
pascal void ExecuteCallback
    (QTCallback cb);
pascal OSErr RemoveCallbackFromTimeBase
    (QTCallback cb);
pascal QTCallback GetFirstCallback
    (TimeBase tb);
pascal QTCallback GetNextCallback
    (QTCallback cb);

```

## Pascal Summary

---

### Constants

---

```

CONST
    {type value}
    clockComponentType          = 'clok';    {clock component}

    {subtype values}
    systemTickClock             = 'tick';    {system tick clock}
    systemMicrosecondClock      = 'micr';    {system microsecond clock}
    systemSecondClock           = 'seco';    {system second clock}
    systemMillisecondClock       = 'mill';    {system microsecond clock}

    {constants for manipulating clock component capability flags}
    kClockRateIsLinear          = 1;          {linear clock rate}
    kClockImplementsCallbacks   = 2;          {clock to implement callback }
                                         { routines}

    {constants to refer to request codes for supported routines}
    kClockGetClockTimeSelect     = $1;        {ClockGetTime}
    kClockNewCallbackSelect      = $2;        {ClockNewCallback}

```

## Clock Components

```

kClockDisposeCallBackSelect    = $3;    {ClockDisposeCallBack}
kClockCallMeWhenSelect        = $4;    {ClockCallMeWhen}
kClockCancelCallBackSelect    = $5;    {ClockCancelCallBack}
kClockRateChangedSelect       = $6;    {ClockRateChanged}
kClockTimeChangedSelect       = $7;    {ClockTimeChanged}
kClockSetTimeBaseSelect       = $8;    {ClockSetTimeBase}
kClockStartStopChangedSelect  = $9;    {ClockStartStopChanged}
kClockGetRateSelect           = $A;    {ClockGetRate}

qtcBNeedsRateChanges          = 1;    {wants to know about rate changes}
qtcBNeedsTimeChanges          = 2;    {wants to know about time changes}
qtcBNeedsStartStopChanges     = 4;    {wants to know when time base start }
                                   { or stop has changed}

{values for callBackType parameter of ClockNewCallBack function that }
{ indicate when a callback event is to be invoked}
    callBackAtTime             = 1;    {at specific time}
    callBackAtRate             = 2;    {when the rate for the time base }
                                   { reaches a specific value}
    callBackAtTimeJump         = 3;    {when a program changes the time value }
                                   { for a time base}

{values for the parameter param1 of ClockCallMeWhen function that indicate }
{ when callback function should be called}
triggerTimeFwd                = $0001; {when current time exceeds trigger time going }
                                   { forward}
triggerTimeBwd                = $0002; {when current time exceeds trigger time going }
                                   { backward}
triggerTimeEither             = $0003; {when current time exceeds trigger time going }
                                   { either direction}
triggerRateLT                 = $0004; {when rate changes to less than trigger value}
triggerRateGT                 = $0008; {when rate changes to greater than trigger }
                                   { value}
triggerRateEqual              = $0010; {when rate changes to equal trigger value}
triggerRateLTE                = $0014; {when rate changes to less than or equal }
                                   { trigger value}
triggerRateGTE                = $0018; {when rate changes to greater than or equal }
                                   { to trigger value}
triggerRateNotEqual           = $001C; {when rate is not equal to trigger value}
triggerRateChange             = 0;    {whenever rate changes}

```



## Data Type

---

```

TYPE
    QTCallback                      = ^CallbackRecord;

    QTCallbackHeader =
        RECORD
            callbackFlags:   LongInt;   {component flags about callback }
                                         { events}
            reserved1:       LongInt;   {reserved}
            qtPrivate:       ARRAY[0..39] OF Byte;
                                         {reserved}
        END;

    QTCallbackFlags    = Byte;

    QTCallbackType     = Byte;

    QTCallbackProc     = ProcPtr;

```

## Clock Component Routines

---

### Getting the Current Time

```

FUNCTION ClockGetTime      (aClock: ComponentInstance;
                           VAR out: TimeRecord): ComponentResult;

```

### Using the Callback Functions

```

FUNCTION ClockNewCallback  (aClock: ComponentInstance; tb: TimeBase;
                           callbackType: Integer): QTCallback;

FUNCTION ClockCallMeWhen   (aClock: ComponentInstance; cb: QTCallback;
                           param1: LongInt; param2: LongInt;
                           param3: LongInt): ComponentResult;

FUNCTION ClockCancelCallback
                           (aClock: ComponentInstance;
                           cb: QTCallback): ComponentResult;

FUNCTION ClockDisposeCallback
                           (aClock: ComponentInstance;
                           cb: QTCallback): ComponentResult;

```

**Managing the Time**

```

FUNCTION ClockTimeChanged    (aClock: ComponentInstance;
                             cb: QTCallback): ComponentResult;

FUNCTION ClockRateChanged    (aClock: ComponentInstance;
                             cb: QTCallback): ComponentResult;

FUNCTION ClockStartStopChanged
                             (clock: ComponentInstance; cb: QTCallback;
                              startChanged: Boolean; stopChanged: Boolean):
                              ComponentResult;

FUNCTION ClockSetTimeBase    (aClock: ComponentInstance;
                             tb: TimeBase): ComponentResult;

```

**Movie Toolbox Clock Support Routines**

---

```

FUNCTION AddCallbackToTimeBase
                             (cb: QTCallback): OSErr;

PROCEDURE ExecuteCallback    (cb: QTCallback);

FUNCTION RemoveCallbackFromTimeBase
                             (cb: QTCallback): OSErr;

FUNCTION GetFirstCallback     (tb: TimeBase): QTCallback;

FUNCTION GetNextCallback      (cb: QTCallback): QTCallback;

```