# 7

# Timer Manager

## 7.1 Introduction

The **timer manager** provides support for timer facilities. The directives provided by the timer manager are:

| Name | Directive Description |
|---|---|
| timer_create | Create a timer |
| timer_ident | Get ID of a timer |
| timer_cancel | Cancel a timer |
| timer_delete | Delete a timer |
| timer_fire_after | Fire timer after interval |
| timer_fire_when | Fire timer when specified |
| timer_reset | Reset an interval timer |

## 7.2 Background

### 7.2.1 Timers

A timer is an **RTEMS** object which allows the application to schedule operations to occur at specific times in the future. User supplied timer service routines are invoked by the **clock_tick** directive when the timer fires. Timer service routines may perform any operations or directives which normally would be performed by the application code which invoked the **clock_tick** directive.

The timer can be used to implement watchdog routines which only fire to denote that an application error has occurred. The timer is reset at specific points in the application to insure that the watchdog does not fire. Thus, if the application does not reset the watchdog timer, then the timer service routine will fire to indicate that the application has failed to reach a reset point. This use of a timer is sometimes referred to as a "keep alive" or a "deadman" timer.

### 7.2.2 Timer Service Routines

The timer service routine should adhere to C calling conventions and have a prototype similar to the following::

```
rtems_timer_service_routine user_routine(
  rtems_id   timer_id,
  void      *user_data
);
```

Where the **timer_id** parameter is the **RTEMS** object ID of the timer which is being fired and **user_data** is a pointer to user-defined information which may be utilized by the timer service routine. The argument **user_data** may be **NULL**.

## 7.3 Operations

### 7.3.1 Creating a Timer

The **timer_create** directive creates a timer by allocating a **Timer Control Block** (TMCB), assigning the timer a user-specified name, and assigning it a timer ID. Newly created timers do not have a timer service routine associated with them and are not active.

### 7.3.2 Obtaining Timer IDs

When a timer is created, **RTEMS** generates a unique timer ID and assigns it to the created timer until it is deleted. The timer ID may be obtained by either of two methods. First, as the result of an invocation of the **timer_create** directive, the timer ID is stored in a user provided location. Second, the timer ID may be obtained later using the **timer_ident** directive. The timer ID is used by other directives to manipulate this timer.

### 7.3.3 Initiating an Interval Timer

The **timer_fire_after** directive initiates a timer to fire a user provided timer service routine after the specified number of clock ticks have elapsed. When the

interval has elapsed, the timer service routine will be invoked from the **clock_tick** directive.

### 7.3.4 Initiating a Time of Day Timer

The **timer_fire_when** directive initiates a timer to fire a user provided timer service routine when the specified time of day has been reached. When the interval has elapsed, the timer service routine will be invoked from the **clock_tick** directive.

### 7.3.5 Canceling a Timer

The **timer_cancel** directive is used to halt the specified timer. Once canceled, the timer service routine will not fire unless the timer is reinitiated. The timer can be reinitiated using the **timer_reset**, **timer_fire_after**, and **timer_fire_when** directives.

### 7.3.6 Resetting a Timer

The **timer_reset** directive is used to restore an interval timer initiated by a previous invocation of **timer_fire_after** to its original interval length. The timer service routine is not changed or fired by this directive.

### 7.3.7 Deleting a Timer

The **timer_delete** directive is used to delete a timer. If the timer is running and has not expired, the timer is automatically canceled. The timer's control block is returned to the TMCB free list when it is deleted. A timer can be deleted by a task other than the task which created the timer. Any subsequent references to the timer's name and ID are invalid.

## 7.4 Directives

This section details the timer manager's directives. A subsection is dedicated to each of this manager's directives and describes the calling sequence, related constants, usage, and status codes.

### 7.4.1 TIMER_CREATE - Create a timer

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_create(
  rtems_name  name,
  rtems_id   *id
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer created successfully |
| **INVALID_NAME** | invalid timer name |
| **TOO_MANY** | too many timers created |

## DESCRIPTION:

This directive creates a timer. The assigned timer id is returned in **id**. This id is used to access the timer with other timer manager directives. For control and maintenance of the timer, **RTEMS** allocates a TMCB from the local TMCB free pool and initializes it.

## NOTES:

This directive will not cause the calling task to be preempted.

### 7.4.2 TIMER_IDENT - Get ID of a timer

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_ident(
  rtems_name  name,
  rtems_id   *id
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer identified successfully |
| **INVALID_NAME** | timer name not found |

## DESCRIPTION:

This directive obtains the timer **id** associated with the timer **name** to be acquired. If the timer name is not unique, then the timer id will match one of the timers with that name. However, this timer id is not guaranteed to correspond to the desired timer. The timer id is used to access this timer in other timer related directives.

## NOTES:

This directive will not cause the running task to be preempted.

### 7.4.3 TIMER_CANCEL - Cancel a timer

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_cancel(
  rtems_id id
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer canceled successfully |
| **INVALID_ID** | invalid timer id |

## DESCRIPTION:

This directive cancels the timer **id**. This timer will be reinitiated by the next invocation of **timer_reset**, **timer_fire_after**, or **timer_fire_when** with **id**.

## NOTES:

This directive will not cause the running task to be preempted.

### 7.4.4 TIMER_DELETE - Delete a timer

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_delete(
  rtems_id id
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer deleted successfully |
| **INVALID_ID** | invalid timer id |

## DESCRIPTION:

This directive deletes the timer specified by **id**. If the timer is running, it is automatically canceled. The TMCB for the deleted timer is reclaimed by **RTEMS**.

## NOTES:

This directive will not cause the running task to be preempted.

A timer can be deleted by a task other than the task which created the timer.

### 7.4.5 TIMER_FIRE_AFTER - Fire timer after interval

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_fire_after(
  rtems_id                          id,
  rtems_interval                    ticks,
  rtems_timer_service_routine_entry  routine,
  void                              *user_data
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer initiated successfully |
| **INVALID_ID** | invalid timer id |
| **INVALID_NUMBER** | invalid interval |

## DESCRIPTION:

This directive initiates the timer specified by **id**. If the timer is running, it is automatically canceled before being initiated. The timer is scheduled to fire after an interval **ticks** clock ticks has passed. When the timer fires, the timer service routine **routine** will be invoked with the argument **user_data**.

## NOTES:

This directive will not cause the running task to be preempted.

### 7.4.6 TIMER_FIRE_WHEN - Fire timer when specified

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_fire_when(
  rtems_id                          id,
  rtems_time_of_day                *wall_time,
  rtems_timer_service_routine_entry  routine,
  void                             *user_data
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer initiated successfully |
| **INVALID_ID** | invalid timer id |
| **NOT_DEFINED** | system date and time is not set |
| **INVALID_CLOCK** | invalid time of day |

## DESCRIPTION:

This directive initiates the timer specified by **id**. If the timer is running, it is automatically canceled before being initiated. The timer is scheduled to fire at the time of day specified by **wall_time**. When the timer fires, the timer service routine **routine** will be invoked with the argument **user_data**.

## NOTES:

This directive will not cause the running task to be preempted.

### 7.4.7 TIMER_RESET - Reset an interval timer

## CALLING SEQUENCE:

```
rtems_status_code rtems_timer_reset(
  rtems_id   id
);
```

## DIRECTIVE STATUS CODES:

| | |
|---|---|
| **SUCCESSFUL** | timer reset successfully |
| **INVALID_ID** | invalid timer id |
| **NOT_DEFINED** | attempted to reset a when timer |

## DESCRIPTION:

This directive resets the timer associated with **id**. This timer must have been previously initiated with a **timer_fire_after** directive. If active the timer is canceled, after which the timer is reinitiated using the same interval and timer service routine which the original **timer_fire_after** directive used.

## NOTES:

This directive will not cause the running task to be preempted.