16

Fatal Error Manager

16.1 Introduction

The **fatal error manager** processes all fatal or irrecoverable errors. The directive provided by the fatal error manager is:

Name	Directive Description
fatal_error_occurred	Invoke the fatal error handler

16.2 Background

The fatal error manager is called upon detection of an irrecoverable error condition by either **RTEMS** or the application software. Fatal errors can be detected from three sources:

- *the executive (**RTEMS**)*
- *user system code*
- *user application code*

RTEMS automatically invokes the fatal error manager upon detection of an error it considers to be fatal. Similarly, the user should invoke the fatal error manager upon detection of a fatal error.

A user-supplied fatal error handler can be specified in the **User Extension Table** to provide access to debuggers and monitors which may be present on the target hardware. If configured, the fatal error manager will invoke a user-supplied fatal

error handler. If no user handler is configured or if the user handler returns control to the fatal error manager, then the **RTEMS** default fatal error handler is invoked. In general, the default handler will disable all maskable interrupts, place the error code in a known processor dependent place (generally either on the stack or in a register), and halt the processor. The precise actions of the **RTEMS** fatal error handler are processor dependent and are discussed in the **Default Fatal Error Processing** chapter of the **C Applications Supplement** document for a specific target processor.

16.3 Operations

16.3.1 Announcing a Fatal Error

The **fatal_error_occurred** directive is invoked when a fatal error is detected. This directive is responsible for invoking an optional user-supplied fatal error handler and/or the **RTEMS** fatal error handler. All fatal error handlers are passed an error code to describe the error detected.

Occasionally, an application requires more sophisticated fatal error processing such as passing control to a debugger. For these cases, a user-supplied fatal error handler can be specified in the **RTEMS** configuration table. The **User Extension Table** field **fatal** contains the address of the fatal error handler to be executed when the **fatal_error_occurred** directive is called. If the field is set to **NULL** or if the configured fatal error handler returns to the executive, then the default handler provided by **RTEMS** is executed. This default handler will halt execution on the processor where the error occurred.

16.4 Directives

This section details the fatal error 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.

16.4.1 FATAL_ERROR_OCCURRED - Invoke the fatal error handler

CALLING SEQUENCE:

```
void volatile rtems_fatal_error_occurred(
  rtems_unsigned32  the_error
);
```

DIRECTIVE STATUS CODES

NONE

DESCRIPTION:

This directive processes fatal errors. If the FATAL error extension is defined in the configuration table, then the user-defined error extension is called. If configured and the provided FATAL error extension returns, then the **RTEMS** default error handler is invoked. This directive can be invoked by **RTEMS** or by the user's application code including initialization tasks, other tasks, and ISRs.

NOTES:

This directive supports local operations only.

Unless the user-defined error extension takes special actions such as restarting the calling task, this directive **WILL NOT RETURN** to the caller.

The user-defined extension for this directive may wish to initiate a global shutdown.