CSR	MASSACHUSETTS INSTITUTE OF TECHNOLOGY CENTER FOR SPACE RESEARCH CAMBRIDGE, MASSACHUSETTS 02139	
REVISION LOG	TITLE: Fatal Error	DOC. NO. 36-53243

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
A	01/16/96	36-476	ALL	New Document	[illegible] 1/23/96

29.0 FatalError (36-53243 A)

29.1 Purpose

The Fatal Error class provides notification that an irrecoverable condition exists and controls an expeditious watchdog CPU reset.

29.2 Uses

Any of the processes or functions may use Fatal Error. Normally, requests for this service are a result of some function encountering a illegal value or condition.

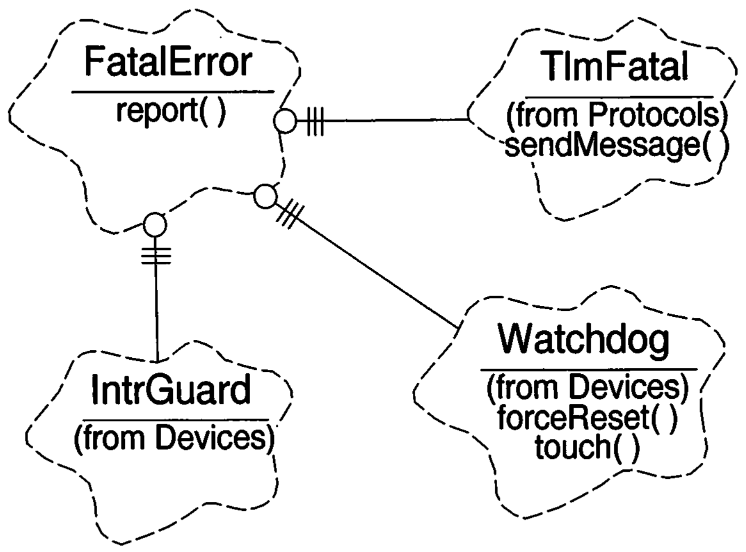
Specifically it provides the following features:

Use 1:: Initiates a panic message which identifies the fault encountered
Use 2:: Forces a system reset.

29.3 Organization

Figure 129 illustrates the relationship between the classes used by Fatal Error.

FIGURE 129. Fatal Error Class Relationships



Fatal Error uses *`Devices`*, and, *`Protocols,`* class categories.

`IntrGuard` - This class is provided by the *`Devices`* class category, and is used to prevent interrupts from interfering with `FatalErrors` activities.

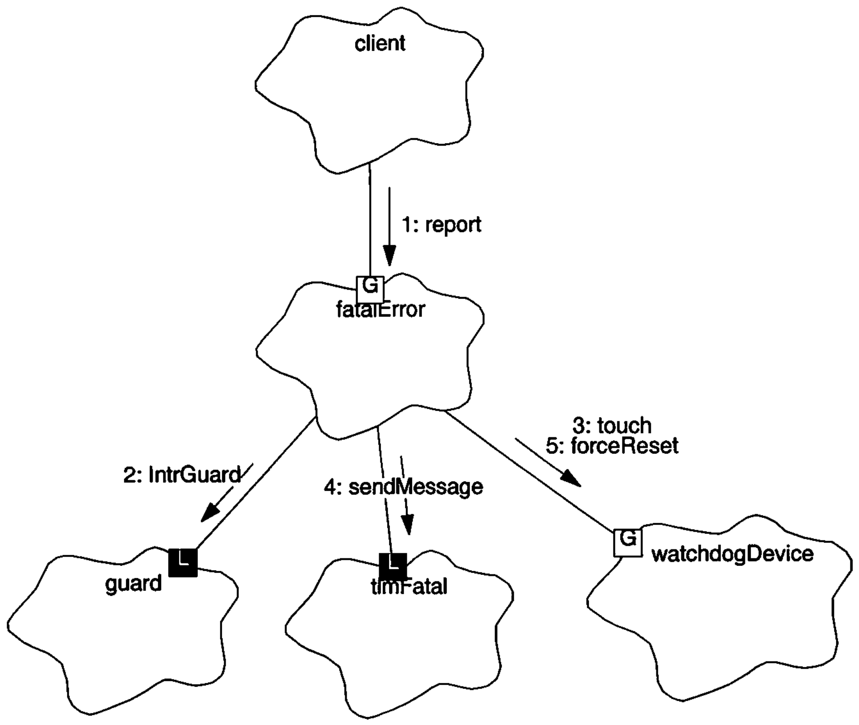
`TlmFatal` - This class is provided by the *`Protocols`* class category and is responsible for insertion of the data into the packet and for initiating delivery of the panic message.

`Watchdog` - This class is provided by the *`Devices`* class category and is responsible for resetting the hardware watchdog timer.

29.4 Scenarios

The `FatalError.report()` 1: may be called by any active process. It is delivered a value identifying the error encountered, and a second argument which provides further information. A call to `FatalError.report()` never returns.

FIGURE 130. Fatal Error Scenario



29.4.1 Use 1: Deliver Panic Message

`FatalError.report()` invokes `IntrGuard.guard` 2: which disables interrupts. `FatalError` will then `touch()` 3: the watchdog providing sufficient time to telemeter the error message. Failure to complete the following steps will result in the watchdog resetting when its regular interval completes since the disabled interrupts will keep the `taskMonitor()` from touching the watchdog.

report() then delivers the information to the **TlmFatal** form using its sendMessage() 4: function which installs the arguments provided by the client into a packet buffer, and hands it off to **TlmManager**.sendPanic() for delivery (not shown). sendPanic() attempts to allow an outgoing message to complete before resetting the telemetry device, handing off the message, and idling for a nominal interval before returning.

29.4.2 Use 2: Handle Watchdog

The **Watchdog**.forceReset() 5: is used to reset that device to the shortest interval, and then busy loops until the CPU is reset.

29.5 Class Fatal Error

Documentation

`FatalError` provides the ability to issue a fatal error telemetry report, then hastens the hardware watchdog's reset of the system.

Export Control:		Public
Cardinality:		1
Hierarchy		
	Superclasses:	**none**
Public Interface		
	Operations:	
		`FatalError()`
		`report()`
Concurrency:		Synchronous
Persistence:		Transient

29.5.1 FatalError()

Public member of:	**FatalError**
Return Class:	**void**
Arguments:	**none**

Documentation

This constructor initializes the `FatalError` instance.

Concurrency:	Sequential

29.5.2 report()

Public member of: **FatalError**

Return Class: **void**

Arguments:

enum Fatal_Code `errorNum`
unsigned `opInfo`

Documentation

`report()` provides the means to control interrupts, deliver a panic message and set the shortest ***`Watchdog`*** interval to immediately reset the CPU.

Semantics

When a client activates `report()`, it disables interrupts, touches the watchdog timer, initiates installation of the arguments provided into the packet using **`TlmFatal`**`.sendMessage()` which hands it off to the ***`Telemetry Manager`*** for delivery. `report()` invokes **`Watchdog`**`.forceReset()` which will cause an immediate watchdog reset.

Postconditions

This function Never returns.

Concurrency: Sequential