FailSafe® System Recovery

What if your system contains compromised memory devices, questionable Flash integrity, a possible hard disk failure and will not boot -- or your system experiences a total operational failure? What if you need to determine the system's current status and update or reinstall all system firmware without replacing any hardware or dispatching a technician to reconfigure or reinstall software? Can it be done? Yes! ZF Micro Devices' ZFx86 System-On-a-Chip contains the primitives to implement a robust process to check and recover your system.

The patented ZF Micro Solutions FailSafe® mechanism is system control at the root, and the ZFx86 by itself can function as the root. This paper describes the ZFx86 FailSafe elements and their operation.

ROOT SYSTEM CONTROL

The FailSafe system's most basic requirement is the ability to control a minimal hardware environment at the root level. Using the ZFx86 with its unique internal functionality makes this possible as the ZFx86 is able to function as the minimal hardware by itself. A few term definitions are necessary to understand the ZFx86 FailSafe sub-system: Pre-Boot Integrity, Fault Detection/Recovery, and the use of Minimal Required Resources.

Pre-Boot Integrity

Before the operating system or an application boots, the system confirms its hardware integrity and validates stored software. Beginning with the BIOS, the ZFx86 validates pre-boot integrity before initiating the full system boot operation and before initializing the system resources. During this process, the ZFx86 uses minimal system resources as everything is assumed suspect until validated.

Operational Fault Detection/Recovery

Once the full system becomes operational, the designed fault detection sub-system is also operational. The fault detection mechanisms detect abnormal system function and then initiate an analysis and recovery mechanism when necessary.

Minimal Required Resources

The key to high quality FailSafe is the system's ability to remain under control with hardware resources held to an absolute minimum. Because we recognize these minimum hardware resources as single points of failure, they directly affect system reliability. The fewer points required, the more robust the design. All systems contain single failure points, and to increase fault resistance, the number of failure points must be reduced causing the probability of system survival to increase. The objective in robust system design is therefore to minimize single points of failure and limit them to elements having an extremely low mean time between failures.

ZFx86 Unique Root Level control

The ZFx86 contains integral features that provide designers with system survivability while still providing complete system control. The ZFx86 addresses the states of pre-boot and full-up operation, because we recognize their potential as single points of failure.

Pre-Boot

The ZFx86 pre-boot function contains three elements: BUR in ROM, BUR Monitor, and SEEPROM Extensions. They allow control of and communications with the ZFx86 without using system resources. Through these, you verify sub-system integrity before a normal boot, or perform post-failure diagnosis without depending on them for operation. Figure 1 shows a generic pre-boot flow:

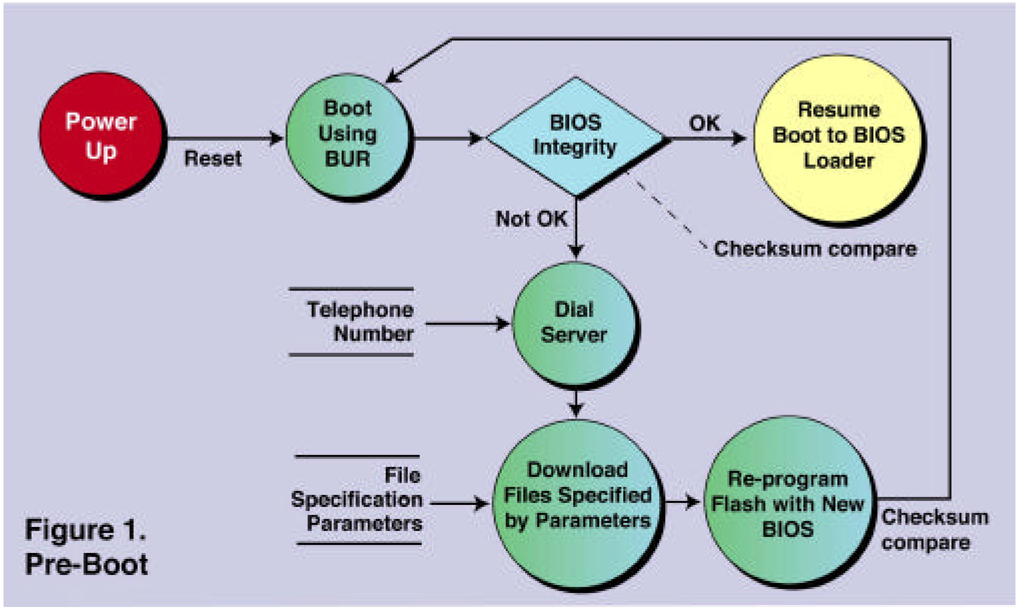

Power Up
Reset
Boot Using BUR
BIOS Integrity
OK
Resume Boot to BIOS Loader
Not OK
Checksum compare
Telephone Number
Dial Server
File Specification Parameters
Download Files Specified by Parameters
Re-program Flash with New BIOS
Checksum compare

Figure 1. Pre-Boot

BUR in ROM

The ZFx86 contains a **B**oot **U**pdate **R**om (BUR) mask programmed in ROM. This code executes at a system reset if an external pin is grounded (either through a jumper setting or when hard wired in the design). This code automatically validates whether or not a serial memory device is connected to the Z-TAG port. The designer defines the serial memory device's contents. If the ZFx86 sees the serial memory device, it then downloads the contents through the Z-TAG port and executes the instructions. During this process, the system's memory is not required.

BUR Monitor

When creating code in the serial memory device mentioned above, the designer can include a command that starts the ZFiX console. This presents a command line interpreter interface to the host machine or a terminal connected to the ZFx86's serial port (COM1). The designer can use this feature to examine and modify the Internal registers, the ISA space (Flash, etc.), and I/O ports using this remote console mode.

BUR Extensions

As described above, the designer programs BUR extensions into the serial memory device, typically a Serial Electrically Erasable PROM (SEEPROM). This device is protected from modification since the Z-TAG port is input only and not capable of sending information to the serial memory. The code stored in the SEEPROM executes the checksum compare shown in the Pre-Boot diagram.

Fault Detection/Recovery

Once a system boots and is in operation, the next FailSafe control layer becomes available. The ZFx86 contains a programmable interval, dual watchdog timer the output of which can be steered depending on the design objectives. In addition, it contains two scratch registers that make it possible to store and retrieve the ZFx86's state at a later time, even after a system reset. Figure 2 shows a generic operation flow:

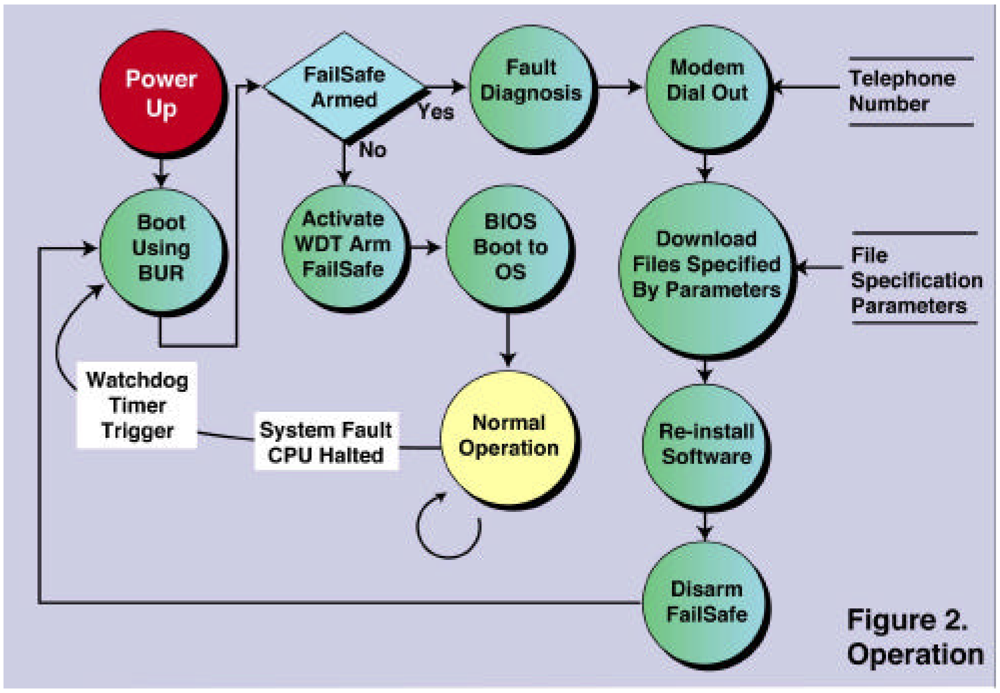
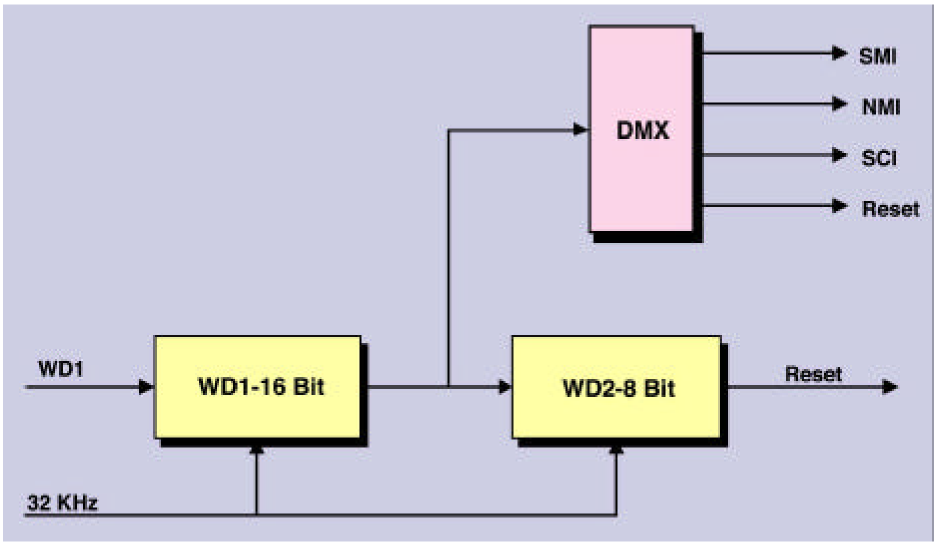


Figure 2. Operation

Dual Watchdog Timer

This diagram shows the first Fault Detection/Recovery element, the Watchdog Timer:



The Watchdog Timer steers the first counter's output to one of four outputs. Each output triggers a different action: the **SMI** and **NMI** interrupts depend upon memory being operational with viable interrupt vectors and handling routines. A hard **reset** might be a secondary course of action if it has been determined that the memory has been compromised. Depending on system requirements, you can program the solutions to these varying scenarios using the SEEPROM (BUR extension) or a higher level application.

Scratch Registers

The scratch register set is the second key element in Fault Detection/Recovery. The two register types behave differently during reset. The first set contains ten persistent registers that survive everything except a power off/on cycle (they are cleared on power-up). The second set includes ten temporary registers that are cleared on power-up or a hard reset.

By accessing these registers, the BUR extensions determine which of the following the system experienced:

- A power-up reset – all registers are cleared
- A hard reset – persistent registers maintain their value, but the temporary registers are cleared
- A soft reset – all registers retain their values

In addition to determining the nature of the reset, the ZFx86 sets a flag that identifies if the watchdog timer caused the reset. Thus, among the two register types and this flag, the BUR extension code can determine the state that existed immediately preceding a normal power-up, or a fault induced reset. These registers also define whether FailSafe is armed, and then after FailSafe determines the cause of the reset, it then prescribes the next step in the recovery process.

External Access

The third Fault Detection/Recovery element needed is the ability to easily construct a design that completes its own external connection in the event of an operational failure. Using the low level control already described, the ZFx86 is able to establish the connection that allows external (and therefore remote) device control using minimal resources. The BUR extension software can communicate through the serial port connected to a host system or a modem. Using a modem, the designer can store an access phone number as part of the recovery code contained in the SEEPROM that can be used to dial a remote connection to a server or monitoring system for diagnostic interaction or simple alarm purposes.

THE MINIMALIST ZFX86

In each of the Fault Detection/Recovery phases, the only hardware required, in addition to the ZFx86, is power, a clock, the SEEPROM, and a connection to the outside world. See Figure 4.

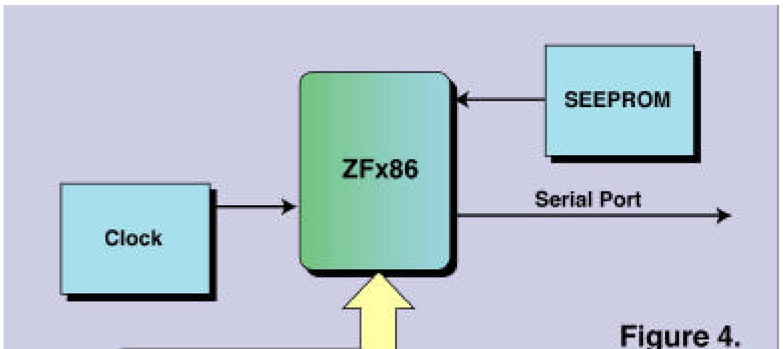


Figure 4. Minimalist System

Once you load the new BIOS image that uses the FailSafe recovery mechanism, the designer can take Fault Detection and Recovery a step further using a significant ZFx86 BIOS feature. This feature allows the designer to include an O/S loader, the O/S, and an application in the same Flash as the BIOS. Thus, the ZFx86 limits a significant portion of normal operation to a highly constrained amount of hardware, inherently providing a highly robust system.