SECTION 18

Crash Handling

18.1 ILLEGAL OPERATION

When the microcode detects an irrecoverable or "can't happen" error, it will crash the system. This process is known as ILLOP after the Illegal Instruction Operation microcode routine that performs it.

ILLOP causes the machine to halt. Further operation in the presence of the irrecoverable error may only worsen the situation or complicate it beyond analysis. Instead, ILLOP will make notes about the error in a crash record stored in the nonvolatile NVRAM on the System Interface Board, and then halt the machine.

ILLOP is a very low level routine. It assumes very little about the state of the machine and it will just halt if it detects that any of its assumptions are wrong. It does not require that any of interrupts, device support, virtual memory, storage allocation, garbage collection, Lisp object support, function calling, or instruction execution be intact. It does assume that the processor is functioning properly; that the registers A-Zero, M-Zero, A-Ones, and M-Ones are set up to contain 0 or -1 as appropriate; that the NuBus is available; and the the NVRAM can be read and written.

After the crash RAM has been written, the crash is indicated by complementing the video sense of the screen. The effect is dramatic. This may fail if the memory interface or the screen interface is not functioning properly but a failure of this operation will not affect the proper recording of the crash in the crash record.

The analysis of crash records is performed by the Lisp crash analyzer. The crash analyzer functions are documented fully in the Explorer Tools and Utilities manual, as are many of the higher level details of crash reporting. This section is intended to be a supplement to that discussion directed at systems programmers. For the final word, consult the Lisp code in SYS:NVRAM; CRASH and the other files in the NVRAM system.

NOTE .

Unless otherwise indicated, all symbols and functions named here are in the SYSTEM package.

18.2 CRASH RECORDING

ILLOP stores some of the machine state in the NVRAM so that the next successful startup can explain the cause of the crash. If the system cannot be successfully started, field service can read the crash reason from diagnostic hardware and/or software. This data is called a crash record.

In order that an unsuccessful attempt to restart the system will not lose the original crash data, crash records for the last few system shutdowns are kept in a circular buffer in NVRAM. Each time the system is started a record is allocated from the buffer. When the system halts, the reason is recorded in the crash record.

NOTE

In order for proper recording of crash information to take place, the structure of NVRAM must first be initialized using the SETUP-NVRAM function. Only then can valid crash records be recorded. Since NVRAM is non-volatile memory, the SETUP-NVRAM function should only need to be run when the system is first installed or after service maintenance has been performed on the System Interface Board.

18.2.1 Crash Record Allocation.

Allocation of a crash record for the current system startup is performed by the microcode during the boot process. It occurs as early in the boot as possible so that useful information can be recorded by ILLOP if the machine crashes during the boot.

Allocation of the crash record is controlled by four 16-bit numbers that are stored at a known place within the NVRAM. These are shown in Table 18-1. All of the allocation registers contain 16-bit byte offsets into the NVRAM, which is accessed using physical memory references. Offsets are expressed in hexadecimal.

NVRAM is actually 8-bit memory mapped into the NuBus address space one-byte-per-32-bit-NuBus-word; hence only multiple-of-four byte addresses are used (eg, 0, 4, 8, #xC, #x10, ...). A 16-bit quantity (such as the crash record allocation registers) is stored with its low order bits in the lowest address and its high order bits in the address 4 higher. For example, #xF4B2 would be stored in NVRAM-CRASH-BUFF-POINTER with #xB2 in NVRAM-BASE + #x90 and #xF4 in NVRAM-BASE + #x94.

NVRAM-CRASH-BUFF-POINTER is the offset into the NVRAM (in bytes) of the beginning of the currently selected crash record. The currently selected crash record describes the current system startup. When the system is running, it points to the record that will be filled in when the system next halts. When the system is not running, it points to the crash record for the last system shutdown.

NVRAM-CRASH-BUFF-REC-LEN is the size of a crash record. It is the amount by which to increase NVRAM-CRASH-BUFF-POINTER to reach the next record.

NVRAM-CRASH-BUFF-LAST is the offset to the beginning of the last crash record in the buffer. This is used by allocation and also when scanning the buffer backward to see the history of shutdowns.

NVRAM-CRASH-BUFF-BASE is the offset to the beginning of the first crash record in the buffer. This is used by allocation to "wrap around" the buffer.

The algorithm to allocate a new crash record, then, is: Add NVRAM-CRASH-BUFF-REC-LEN to NVRAM-CRASH-BUFF-POINTER. The result is the new NVRAM-CRASH-BUFF-POINTER. If the new NVRAM-CRASH-BUFF-POINTER is greater than NVRAM-CRASH-BUFF-LAST then it should be reset to NVRAM-CRASH-BUFF-BASE.

The algorithm for finding the previous crash record is: Subtract NVRAM-CRASH-BUFF-REC-LEN from NVRAM-CRASH-BUFF-POINTER. If this is less than NVRAM-CRASH-BUFF-BASE, it should be set to NVRAM-CRASH-BUFF-LAST.

Table 18-1 Crash Record Allocation Registers

NVRAM-BASE plus (hex)	Allocation Register
80	NVRAM-CRASH-BUFF-FORMAT-PROCESSOR
88	NVRAM-CRASH-BUFF-FORMAT-REV
90	NVRAM-CRASH-BUFF-POINTER
98	NVRAM-CRASH-BUFF-REC-LEN
A0	NVRAM-CRASH-BUFF-LAST
A8	NVRAM-CRASH-BUFF-BASE

18.2.2 Crash Record Contents.

The crash record format is shown in Table 18-2. The templates for the crash table, for the rest of NVRAM, and for other information used by crash record support can be found in the file SYS:UCODE;LROY-QDEV. The list CRASH-RECORD-OFFSETS contains a list of symbolic names whose values are these offsets.

Table 18-2 Crash Record Format

NVRAM-CRASH-BUFF-POINTER plus (hex)	Contents
	General information
0	Progress field. Indicates how far into the boot process the system progressed before halting.
	Load information
4	Disk controller slot number
8	Disk device number for microload
C	Disk device number for Load Band
10	Microload name (4)
20	Load Band name (4)
30	Microload version (2)
38	Load Band version (2)
40	Load Band revision (2)
	Date and time information
48	Month of boot
4C	Day of boot
50	Year of boot
54	Hour of boot
58	Minute of boot
5C	Month of crash
60	Day of crash
64	Year of crash
68	Hour of crash
6C	Minute of crash
	Flags field
70	Report Flags
	Shutdown information
74	Halt Location (2)
7C	Halt Kind
	Saved Registers
80	contents of M-1 (4)
90	contents of M-2 (4)
A0	contents of MD (4)
B0	contents of VMA (4)
C0	contents of M-FEF (4)
D0	contents of UPC-1 (2)

D8	contents of UPC-2 (2)
E0	Location counter (4)
F0	contents of M-T (4)
100	Length of crash record

The progress field indicates how far into the boot process the system progressed before crashing. Currently supported values for this field are listed in the variable CRASH-RECORD-PROGRESS-CODES.

The load information fields are initialized during crash record allocation by the microcode. They reflect the load information saved by the boot process.

The boot time is written to the crash record by a function on the WARM-INITIALIZATIONS-LIST after the system time base has been initialized. The crash time fields are updated about every five minutes from Lisp by the TM-Update process so that they will accurately reflect the time if the machine halts.

The report flags are used to store information about which crash records have been logged to a crash-log file by the crash analyzer (see section on crash analyzer below). It is also contains a flag that is set if this boot was a warm boot so that information can be displayed in the crash record. Crashs that occur after a warm boot or warm boot attempt are often caused by problems in the warm booted environment, and hence are not as interesting as crashes that occur in a cold booted environment.

The halt kind field indicates the type of the last shutdown. The field is initialized to the System Boot state by the crash record allocation microcode, then updated during ILLOP to reflect the crash kind. Possible values for halt kind are:

* System Boot. The last shutdown was caused by a cold boot sequence or by a warm boot sequence during normal operation (that is, not a warm boot initiated after the machine crashed). Note that since the crash record allocation routine is called from warm-boot, the system can be warm booted after an abnormal shutdown and still retain all the crash record information from that shutdown. However, if the machine was in a hung state when warm or cold booted, ILLOP processing will not be done, and the shutdown will be reported in this category. META-CONTROL-META-CONTROL-C can be used to force a crash from a hard run, and therefore produces a crash record.

* Microcode Halt. This indicates that the last crash was called from the microcode when it detected an unrecoverable error condition. In this case, ILLOP stores the micro-pc address from which ILLOP was called in the halt address field. This micro-pc is later looked up in the crash table database by the crash analyzer in order to provide the user with a text description of the microcode crash reason. See the section on the crash analyzer, below.

* Hardware Halt. This halt kind is currently unsupported. All detectable hardware halt conditions (such as memory parity errors, illegal page faults, and so forth) currently fall into the microcode halt category.

* Lisp Halt. The last halt was called by Lisp through the si:%crash function. In this case, a Lisp crash code (which is one of the arguments to %CRASH) is stored in the halt address field. The second argument to %CRASH (an object) is stored in the M-1 field. Currently, the only valid Lisp crash code is 0 which indicates a normal system shutdown called from Lisp. This code is seen when the system was halted by a user-initiated call to either SHUTDOWN or SYSTEM-SHUTDOWN.

The saved registers fields contain the values of the indicated processor registers when ILLOP was called. Their values can appear in the microcode crash descriptor text reported by the crash analyzer, and in any case their values are labeled and displayed in the crash analysis report.

18.2.3 The Crash Table.

The crash analysis database consists of the table of crash codes and crash descriptions produced by the micro assembler. This table is kept in the file SYS:UBIN;<ucode-name>.CRASH#nnn where nnn is the microcode revision number and ucode-name is obtained by looking up the microcode name associated with the current value of MICROCODE-TYPE-CODE in the *MICROCODE-NAME-ALIST*.. The crash analyzer loads the appropriate version of this file into memory when a microcode crash is being reported. The crash table file contains a complete list of all current crash descriptor texts.

Note that there may be some microcode crashes that will not have a description in the crash database. This fact will be reported by the crash analyzer. The crash micro-pc for such crashes is valid, however, and can be used to determine the path taken to ILLOP.

The crash table file is generated by the microassembler. At points in the microcode where ILLOP may be called a CRASH-TABLE pseudo-op is generated. From this a Crash Table Entry (CTE) is generated and added to the list of CTEs that make up the crash table file. The format of a CRASH-TABLE pseudo-op and the resulting CTE are as follows:

```
Ucode CRASH-TABLE pseudo-op:
----------------------------
General form:
   (CRASH-TABLE <format string> <one or more format args>)
Example:
   (CRASH-TABLE "Nasty data type ~a read from address ~x"
      (Q-DATA-TYPES (LDB %%Q-Data-Type MD))  VMA)

Crash Table Entry (in .CRASH file):
-----------------------------------
General form:
   (<Micro PC> <format string> <one or more format args>)
Example:
   (34174 "Nasty data type ~a read from address ~x"
      (Q-DATA-TYPES (LDB %%Q-Data-Type MD))  VMA)
```

The first element of the CTE is the microcode PC address from which the call to ILLOP was made. The second element is usually a string. In this case we essentially do the following to format the crash description.

```
(APPLY #'FORMAT <stream> (CDR <cte>))
```

This works because the saved register values in the crash record are bound to special variables named VMA, MD, M-FEF, etc., at the time this FORMAT is executed.

If the second element of a CTE is a symbol, we look for the REPORT property on its property list. If present, it will be a report function to run.

Note that the file SYS:NVRAM;ANALYSIS-FUNCTIONS contains a number of crash-description formatting routines that have very detailed knowledge of the contents of the saved microcode registers. These routines must be kept in synch with current microcode usage, just as with the error handler files.