SECTION 2

Bootstrap Loading

2.1 INTRODUCTION

Bootstrap loading the Explorer system involves several stages and several types of loads. While the remainder of this document is involved with system operation, bootstrap loading occurs before the system is operational and is, therefore, quite different from what follows in later sections.

2.2 TYPES OF LOADS

There are four type of loads:

* Power on / System reset. These are exactly the same except for the way they are initiated. A power on boot is performed when power is cycled on the main chassis. A system reset is performed by issuing the boot key chord META-CTRL-META-CTRL-ABORT. Much goes on during this type of load, but when the LISP system is being loaded, the primary software entity that gets loaded is the "primitive" microcode (from now on this will be referred to as the "primitive"). Microcode is loaded from a microload band (also known as a microcode partition) on secondary storage into the writable control store (WCS) of the processor. The system microcode in turn loads the system (the LISP system software) from a load band on secondary storage.

* Cold boot. This is performed by issuing the boot key chord META-CTRL-META-CTRL-RUBOUT. This causes the Explorer to reload the current system microcode and the current system load band from secondary storage. The previous environment is lost.

* Warm boot. This is performed by issuing the boot key chord META-CTRL-META-CTRL-RETURN. This causes the Explorer to restart the system code in such a way as to preserve the previous environment. Neither the system microcode nor the system load are re-read from secondary storage.

* Menu boot. This is performed by issuing the boot key chord META-CTRL-META-CTRL-M. This causes the Menuboot microload (see description of Menuboot in the Power on / Cold boot description below) to be loaded from the same unit that the current system load was loaded from. Menuboot is then executed as described below.

Note that each of the four boot key chords described above will cause the Explorer processor to take a true hardware trap (not just a polled event) to the appropriate point in microcode. So even if the processor is stuck in a loop, it should respond to any of the four chords. An additional key chord, META-CTRL-META-CTRL-C, can be used if the system appears to be in an infinite loop. This causes the system to crash and to record the currently running Lisp function.

If the system microcode in Writable Control Store (WCS) has been violated, then the Cold boot, Warm boot, and Menu boot may not function correctly. In this case you have no choice but to cycle power or do a System reset.

2.3 POWER ON / SYSTEM RESET

Both of these actions cause a NuBus reset which forces all boards that have a self-test to perform it. A NuBus reset causes the Explorer processor to enter its boot ROM based Microcode at a fixed location.

The first action it takes is to perform the processor self-tests.

Then all internal memories and registers are cleared.

If the Explorer I processor fails self-test, it will crash with a light code #x89. The Explorer II processor reports individual selftest numbers as they are run, with the failing test number left on the lights in case of failure. (See the light codes table.)

System Test - if the processor passes its self-test, it proceeds to the next phase, System Test and Boot.

1. First the processor determines whether it is to be the System Test & Boot Master (STBM) or if some other processor will be. In a single processor system, the Explorer will of course be the STBM. In a multi-processor environment, the processor in the lowest numbered slot that is an STBM candidate and has passed

self-test (within 10 seconds) is the STBM.

If the Explorer processor is not the STBM, it enters Secondary mode and waits for the STBM to post an event to awaken it. It then performs secondary booting operations.

2. Assuming the Explorer is the STBM, it performs the following actions:

NOTE

All searches for resources are performed by starting with slot 0 and looking at boards in successively higher numbered slots until slot 15 has been checked. For each slot, the STBM first checks for a value of #xC3 in the ID character of the boards Configuration ROM (see paragraph on the Configuration ROM). This indicates the board has a valid Configuration ROM. Next, check that the CRC in the configuration ROM is correct. Then the Resource Type field in the Configuration ROM of each board is examined to see if the board contains the desired resource.

a. Search for NVRAM. If the NVRAM bit of the Resource Type field is set, then the board contains NVRAM. The Explorer checks the CRC on the contents of the NVRAM (Explorer I does not) and the format generation number to verify it contains a value of #x01. Upon finding a valid NVRAM, the STBM extracts pointers to a Monitor, Keyboard, and Load Source for later use. See the paragraph on NVRAM for format details.

b. Find a monitor. If a valid NVRAM was found, then the monitor slot and unit numbers from NVRAM are used and the monitor is validated by first checking that the monitor bit in the Resource Type field is set, and then by issuing the Initialize- Monitor device driver call. If this completes successfully the monitor has been found. If there was no valid NVRAM, or if the Initialize-monitor call failed, then the processor searches for a board with the monitor bit on in the Resource Type field of the configuration ROM. If it finds one it calls the Initialize-monitor function of the device driver

on that board. If the call completes successfully, the monitor has been found. If no monitor can be found, the processor attempts to perform a default boot.

c. Display the message "Slot S TESTING SYSTEM", where S is the slot number of the processor board, on the monitor.

d. Find a memory board. The processor searches for a board with the memory bit set in the Resource Type field. When it finds one, it validates it by running the Interface Diagnostic, located in the ROM on that board. The diagnostic is run with messages disabled. If it passes, that memory board is used. Otherwise, the processor searches for another memory board. It is assumed that every memory board will have at least two megabyte of memory. If no valid memory board can be found, the message "ERROR: NO GOOD MEMORY FOUND" (Explorer I = "ERROR: 00000004") is displayed on the monitor, and the processor crashes with a light code (Explorer I = #x8A; Explorer II = #x74)

e. If a memory board was found, the processor then performs Chassis Testing where it performs the following actions on each successive board in the system, starting with slot 0 and ending with slot 15. If any test fails, the rest of the tests for that board are skipped.

   - Tests whether a board is present in the slot (if no NuBus timeout is received when reading the configuration ROM, then a board is present). If the slot is empty, go to next slot.

   - Displays "Slot S" on the monitor, where S is the slot number of the board under test.

   - Tests whether the Configuration ROM contents are valid: ID character = #xC3 and CRC verifies. If not, display "ROM" ("TESTS FAILED" displayed later).

   - If the ROM flags in the Configuration ROM indicate that the board performs a self-test, then wait up to 20 seconds for the self-test to complete. The board will reset a bit in the onboard Flag Register (see paragraph on Configuration ROM) when its self-test is complete. Check the Flag Register for a self-

test failure, and if one occurred, display "SELF" ("TESTS FAILED" displayed later).

- If the ROM flags in the Configuration ROM indicate that the board participates in NuBus tests, command it to do so and check the results. If a failure is detected, display "NUBUS" ("TESTS FAILED" displayed later).

- If the Configuration ROM Diagnostic Offset field is not = #xFFFFFF, execute the boards Interface Diagnostic.

- If all tests for a slot have passed, then turn off the slot Test LED in the Configuration Register (see paragraph on the Configuration ROM), and display "passed", otherwise display "TESTS FAILED".

f. Find a keyboard. If a valid NVRAM was found, then the keyboard slot and unit number from NVRAM is used and the keyboard is validated by first checking that the keyboard bit in the Resource Type field is set, and then by issuing the Initialize- keyboard device driver call. If this completes successfully the keyboard has been found. If there was no valid NVRAM, or if the Initialize-keyboard call failed, then the processor first attempts to initialize a keyboard at the same slot and unit as the system monitor currently in use (not supported by Explorer I). If that also fails then it searches for a board with the keyboard bit on in the Resource Type field of the configuration ROM. If it finds one it calls the Initialize-keyboard function of the device driver on that board. If the call completes successfully, the keyboard has been found.

If no keyboard can be found, the processor performs a default load using the boot device slot and unit from NVRAM. If there was no valid NVRAM, the processor searches for a boot device by first searching for a slot that has the boot source bit (Explorer I: and not the LAN bit) in the Resource Type field set, and then using the lowest numbered unit at that slot. The microcode specified as default in the partition table are loaded.

Initial Menu - if a keyboard was found, the processor sounds a tone and displays the Initial Menu: "D=Default load, M=Menu load, R=Retest, E=Extended tests :" If no key is pressed within approximately 15 seconds and no boards have failed Chassis Test, the processor attempts a default load of the MCR partition marked as default on whichever load source is determined to be the default load source. Pressing "D" or "RETURN" also results in a default load.

1. Pressing "R" causes the Chassis Testing phase, described above, to be repeated.

2. Pressing "E" causes the Chassis Testing phase to be repeated, but each Interface Diagnostic is run in "extended mode". In extended mode, an interface diagnostic will display the board identifier, the part number, the name of each test it runs on the screen, and an indication of whether each test passed or failed. Additional testing may also be performed. For example, the memory board diagnostic tests all of memory in extended mode, but not in normal mode.

3. Pressing "M" causes the processor to go into the menu boot sequence, described below.

4. At this point there are a number of "hidden options" in addition to those already mentioned. They are intended for use by expert users such as system managers and maintenance personnel.

   a. Pressing "S" tells the processor that you want to do a default boot, but you want to use a boot unit other than the default unit. The processor will then present the device selection menu (described below), but will perform the boot as soon as you have selected the device.

   b. Pressing "G" tells the processor that you want to load GDOS, the diagnostic operating system. The processor will then present the device selection menu (described below) but will boot GDOS as soon as you have selected the device.

   c. Pressing "N" means that you want to type in the names of microcode and load bands to be booted. The processor will then prompt you for each of these names. You must type these names exactly as they are displayed by print-disk-label; the processor will not convert lower case to upper case automatically. The shift key or caps lock key will give you upper case. The rubout key will let you correct mistakes. Once you have typed in a name, the return key indicates you are

ready to proceed. When you have entered both names, the processor presents the device selection menu (described below). As soon as you select a device, the processor will boot the microcode and load band you typed in.

d. Pressing "F" (not supported on Explorer I) tells the processor that you want to load FDOS, the Factory version of the diagnostic operating system. The processor will then present the device selection menu (described below) but will boot FDOS as soon as you have selected the device.

e. Pressing "!" (not supported on Explorer I) causes the processor to enter a Debug utility menu.

Default Boot. After D is selected for Default Boot, the processor attempts to load from either the default load source specified in NVRAM or the first unit found by searching all slots. A "waiting" message is displayed until the default load source is ready, then the STBM interprets the Device Driver in the load source interface board configuration ROM to load the first MCR partition with the default bit set. Under the Release 3.0 the MCR named PRIM is used for an Explorer I, BOOT for an Explorer II.

Both BOOT and PRIM are primitives and the term "primitive" refers to both. The "primitive" has three primary functions which are:

1. The "primitive" co-ordinates the booting of a machine with multiple processors. Since the Explorer does not currently have multiple processors, this is not done in the "primitive" even though it has a skeleton design for such a function.

2. The "primitive" performs slave device downloading. Downloading can be thought of as patching the ROM on a controller board such as a disk controller board. If there is a software bug in the controller board ROM, it can be fixed by loading new software into a RAM area on the controller board and executing out of the RAM area rather than the ROM area.

3. The "primitive" loads the Lisp MCR code into the processor.

To do all of the above, the "primitive" uses a special type of partition called a Configuration partition. There may be several configuration partitions on the default disk. The default configuration partition is used by the "primitive".

A configuration partition has several entries in it, which are the name of the Lisp MCR, the name of the Lisp Load band, the name of the download software partition, and other information reflecting the machines configuration. If the default configuration reflects an erroneous machine configuration an error will occur. Later in this section there will be a description of the configuration partition and the algorithm used by the "primitive".

It should be noted that on an Explorer I PRIM and BOOT are identical but separate software partitions. On the Explorer II the single partition BOOT takes on both roles as "primitive" and menuboot. Both Menuboot and the "primitive" assume 2MB or greater memory boards.

Menu Boot - If you entered an "M" at the Initial Menu the processor then presents the device selection menu (the header is "AVAILABLE LOAD DEVICES"). This menu lists the available load devices and asks you to select one. After you select a device, the processor then loads a microcode band called "BOOT" from the device you selected. This is Menuboot, which then takes over the remainder of the boot process. It should be noted that this is the point where the processor leaves ROM and enters code (Menuboot) that has been downloaded into WCS. This is important because if for some reason Menuboot does not exist on the device you selected (or if it has been wiped out), this step will not work. If Menuboot is non-existent, the message "MICROLOAD NOT FOUND" will be displayed. If Menuboot has been wiped out, the message "BAD MICROLOAD FORMAT" will be displayed. These messages can also occur on any other microload attempt.

The first thing Menuboot does is to present the menu "L=LISP load, M=Multi-unit load, D=Diagnostic load, P=Print device label, C=Configuration Boot"

1. To do a Configuration Boot you enter an C or simply press RETURN since Configuration Boot is the default. After a C is pressed a menu of load devices appear. The operator is to select the desired device. After the device is selected, a list of configuration partitions appear on the screen. The operator then selects which configuration partition is to be used in the booting process.

2. To do a Lisp load enter "L" . The processor presents a

menu of available load bands on the device previously selected. Asterisks may appear by some of the load bands. The asterisks should be ignored and the user should select the desired load band.

Once you have selected a load band, you will be presented with a menu of available microcode bands. Again, asterisks may appear by some microcode bands. The asterisk may be ignored as above. The microcode that is preferred by the load band you previously selected will be named in a header message above the microcode menu. The preferred microcode is simply the one that the load band was saved with. You can generally use microcodes other than the preferred one, but the system will have to load the error table over the network. Once you have selected a microcode partition, the processor will load that microcode and pass it the name of the load band to be used.

3. If you select "M" the processor will present you with the the device selection menu before each partition menu. This allows you to select the system microcode and and system load band from different devices.

4. If you select "D" the processor will present you with the device selection menu (described above). Once you select a device the processor will display a menu of available diagnostic microloads for that device. Selecting one will cause that microcode to be loaded and executed.

5. If you select "P" the processor will present you with the device selection menu again (see above). When you select a device the processor displays the list of partitions on that device, similar to what is displayed by a (print-disk-label) form in LISP.

2.4 CONFIGURATION ROM

A machine based on the NuBus architecture has up to 16 NuBus slots. The Explorer has seven. Each slot may contain one board. Each board contains a configuration ROM. The configuration ROM is located at the highest NuBus physical addresses allocated for each slot (i.e. the ROM ends at address FsFFFFFC, where s is the slot number of the board) with data stored one byte per NuBus word.

The Configuration ROM fields that are of concern to the booting

process are as follows:

Resource Type field - This is a one byte field, located at address FSFFFF00. Each bit that is set to one indicates a resource that the board contains.

Bit 0 Memory - board contains memory that may be used during booting. This memory will start at FS000000 and be at least one contiguous megabyte long.

Bit 1 Boot source - board is a controller (e.g. disk, tape, LAN) that has a unit(s) that may be used as a load device. This board contains a load device driver.

Bit 2 LAN - board contains a Local Area Network (LAN) controller that may provide a load device via the network This board may contain a load device driver.

Bit 3 Monitor - board has a unit that can be used as the system monitor during the boot process. This board contains a monitor device driver.

Bit 4 Bootable processor - board is capable of performing the standardized functions of a "secondary" processor to an STBM during system booting operations.

Bit 5 Keyboard - board has a unit that can be used as the system keyboard during the boot process. This board contains a keyboard device driver.

Bit 6 NVRAM - board contains non-volatile RAM that may contain system test and boot default parameters. If this bit is set the three byte board relative offset to the NVRAM is stored in locations FSFFFEF4 - FSFFFEFC (one byte per word) and the log 2 of the NVRAM size is stored in location FSFFFEF0.

Identification character - This is a one byte field, located at address FSFFFF04. If it contains the value #xC3, then the configuration ROM contains valid data, otherwise it does not. This provides a way for "foreign" boards to exist in the system without confusing the STBM.

ROM Flags - a one byte field at location FSFFFF10 which contains the following flags (and several others which do not concern us here):

Bit 0 A one indicates the board does self-test.

Bit 1 A one indicates the board will participate in NuBus tests.

Bit 2 A one indicates the board is capable of being an STBM.

Flag Register Offset - a three byte field at locations FSFFFF14 - FSFFFF1C (one byte per word) containing the board relative offset to the Flag Register. The Flag Register is a one byte field containing the following flags:

Bit 0 A one indicates self-test is still in progress.
Bit 1 A one indicates the board failed self-test.
Bit 2 A one indicates that a peripheral or subsystem controlled by this board failed test.

Diagnostic Offset - a three byte field at locations FSFFFF20 - FSFFFF28 containing the board relative offset to the Diagnostic Engine code that makes up the interface diagnostic. A value of >FFFFFF indicates there is no interface diagnostic.

Device Driver Offset - a three byte field at locations FSFFFF2C - FSFFFF34 containing the board relative offset to the Diagnostic Engine code making up the boards device driver(s). There must be one device driver for each of the following bits that are set in the Resource Type field: boot source, monitor, keyboard.

Configuration Register Offset - a three byte field at locations FSFFFF38 - FSFFFF40 containing the board relative offset to the Configuration Register, which is a 8 bit field which contains (among other information) the following:

Bit 0 Reset - writing a one resets the board
Bit 1 NuBus Master Enable - a one enables the board to read and write via the NuBus (0 disables the boards Nubus interface)
Bit 2 Test LED - A one turns on the red LED on the board. A zero turns it off.
Bit 3 Test - when set to one during system testing requests the board to perform its NuBus Test.

CRC Signature - A 2 byte field at locations FSFFFFB8 - FSFFFFBC that contains the CRC value computed over the boards ROM. The ROM size is stored in location FSFFFFB4.

2.5 THE STBM AND DEVICE INDEPENDENCE

One of the goals of the STBM is to allow the ROM code on each board to be independent of the other boards in the system. For example, it must be possible to replace the disk controller board with one that provides a different interface to the processor, but not to have to change the processors boot ROMs. This is accomplished by having a boot source device driver (DDR) in the ROM on the disk controller. The DDR provides the processor with a generic interface to the device. The DDR is written in a

processor independent, interpreted language, called Diagnostic Engine code, and is executed by the processor. Since it is interpreted, the processor must have a Diagnostic Engine Interpreter in its ROM. Being processor independent means the disk controllers ROM does not have to change if the processor is replaced with one of a different type. It is called Diagnostic Engine code because the scheme was originally developed to allow processor independent diagnostics to be written. Most boards in the system have an Interface Diagnostic written in Diagnostic Engine code in their Configuration ROM.

There are three type of DDRs: load source, monitor, and keyboard. They provide the processor with a generic interface to the three peripheral resources it requires to perform system testing and load. The monitor and keyboard allow the processor to communicate with the operator, and the load source provides a source from which to load code. Load source interfaces currently exist for disk, tape, and LAN, and the processor-to-boot-source interface is generic enough to support virtually any kind of device that is capable of loading code. DDRs are intended to be used for bootstrap loading, not for normal system use.

2.6 NVRAM FORMAT

The information in NVRAM is accessed as though it were stored one byte per word. The first part of NVRAM contains system default configuration information, which is accessed during the boot process:

base addr + #x00 = STBM Monitor unit number LSB byte	Binary
base addr + #x04 = STBM Monitor unit number MID byte	Binary
base addr + #x08 = STBM Monitor unit number MSB byte	Binary
base addr + #x0C = STBM Monitor slot number (FF = none)	Binary
base addr + #x10 = STBM Keyboard unit number LSB byte	Binary
base addr + #x14 = STBM Keyboard unit number MID byte	Binary
base addr + #x18 = STBM Keyboard unit number MSB byte	Binary
base addr + #x1C = STBM Keyboard slot number (FF = none)	Binary
base addr + #x20 = Boot source unit number LSB byte	Binary
base addr + #x24 = Boot source unit number MID byte	Binary
base addr + #x28 = Boot source unit number MSB byte	Binary
base addr + #x2C = Boot source slot number (FF = none)	Binary
base addr + #x30 = NVRAM format generation number. Equal 01 for all NuGeneration devices	Binary
base addr + #x34 = NVRAM format superset revision number.	Binary
base addr + #x38 = NVRAM CRC LSB byte	Binary

base addr + #x3C = NVRAM CRC MSB byte Binary

The NVRAM contains additional information in a specific generic format so that it can be accessed by any type of processor. The format is described in detail in the NuBus System Architecture Specification.

2.7 ERROR CODES AND MESSAGES

For the most common error conditions, textual error messages are displayed. However, in some cases, the STBM ROM code and Menuboot will display hexadecimal error codes.

2.7.1 STBM ROMS - Error Codes.

ERROR: 00000002 - Load device offline or not responding.
The device is powered down or is not connected.

ERROR: 00000003 - Load device error.
The load device experienced an unrecoverable error.

ERROR: 00000004 - Processor could not find a memory board that passed tests. The processor checks the following when looking for a memory board: Look for the value #xC3 in Configuration ROM ID character.
Look for memory bit set in Configuration ROM Resource Type field.
Make sure CRC in configuration ROM is correct. Run the Interface Diagnostic in the board's ROM and check that it had ROM and check that it had no failures.

DEVICE ERROR: 00000005 - Unexpected NUBUS error.
The processor was executing diagnostic engine code in a device driver in the NuBus Peripheral Interface (NUPI) or SIB ROM when an unexpected NUBUS error occurred.

DEVICE ERROR: 0000006 - Command timeout.
The NUPI device driver issued a NUPI command block and the NUPI did not set the complete bit in the status field. The minimum timeout value is 10 seconds. If the disk label is messed up, it could possibly cause this problem. The NUPI could also be faulty. If the NUPI considers a command block to be invalid, it will exhibit this failure mode. The last NUPI

command block that the NUPI device driver issued is located at location FS00C000, where S is the slot number of the memory board in the lowest numbered slot.

DEVICE ERROR: 00000009 - Network down.
The Ethernet is disconnected, shorted, or open.

DEVICE ERROR: 0000000A - Invalid unit number for the load device.

DEVICE ERROR: 0000000B - Ethernet board failed to initialize.

DEVICE ERROR: 00000010 - Bad DEI instruction header.
A board was found with a valid configuration ROM, but the Diagnostic Engine code that the Diagnostic offset or Device Driver offset in the configuration ROM points to has an invalid header. This possibly means that the ROM is bad.

DEVICE ERROR: 00000011 - Invalid DEI request.
The ROM on the board is good, but a request was made that could not be handled by that board. (e.g. a boot request was given to the monitor). This probably means that the contents of NVRAM are invalid. Try doing a menu boot, specifying the boot unit. Once the system is booted type in (si:setup-nvram) to the LISP Listener.

DEVICE ERROR: 00000012 - Diagnostic Engine code instruction space (ispace) problems.
The processor found an invalid instruction when trying to execute Diagnostic Engine code out of the ROM on one of the boards. This could happen when executing a diagnostic or a device driver. This possibly means the ROM is bad.

DEVICE ERROR: 00000013 - Diagnostic Engine code data space (dspace) problems.
The processor found one of the following problems when trying to execute Diagnostic Engine code out of the ROM on one of the boards: stack overflow, stack underflow, or dspace variable out of range. This could happen when executing a diagnostic or a device driver. This could be due to a bug in the code being executed, or the ROM could be bad.

DEVICE ERROR: 60000000 and above - NUPI command status.
These are errors that the NUPI device driver

passes back from the status field of the NUPI command block. See the NUPI Hardware Specification.

2.7.2 Menu Boot and Primitive - Error Codes and Error Messages.

ERROR: 00000014 - Device access error. The NUPI returned bad status.

ERROR: 00000015 - Invalid label. The first word of block 0 did not contain "LABL".

ERROR: 00000016 - Invalid partition table. The first word of the partition table did not contain "PRTN".

ERROR: 00000017 - No available microloads. There were no Explorer microcode partitions in the partition table.

2.7.2.1 Menuboot and Primitive Error Messages.

Message	Meaning
Warning: No Microcode Partitions on Device	Device selected in a Lisp load or device specified in a configuration partition had no microcode.
Warning: No Configuration Partition on Device	Device selected in a Configuration Boot had no configuration partition on it.
No Default Configuration Partition	On default boot, the "primitive" could not locate a default configuration partition.
Unable to Read Device	An error occurred when trying to read the load device.
Invalid Slot or Unit Number in the Configuration Partition.	The portion of the configuration partition containing the disk slot and unit number for the Lisp MCR is invalid.
Bad Load Partition or Load Device	The entry portion of the configuration partition containing the information about the load band has invalid information in it.
Currently Executing CPU is not in Configuration	A configuration partition was used which did not have a valid entry for the Explorer processor board.
Cannot Download Device	When trying to download a , device, one of four problems occurred. First, the part number in the entry field may not match; second, there is a problem with the disk slot or unit number specified; third, there is a problem in matching the CPU type to the board type in the configuration partition; and last, there is no match on the download partition name.
Unable to Read the	This means the data in the

Partition Table	partition table is not set up correctly. To be set up correctly the following must be true: There must be a default entry named LABEL with partition type of Volume Label and CPU type of generic. There also must be a default entry named PTBL with partition type of Partition Table and CPU type of generic.

2.8 LIGHT CODE TABLE

In some cases the processor cannot proceed and cannot display a message. In these cases the processor will crash and display a code in the amber colored light-emitting diodes (LEDs) located on the processor board. These can be viewed by opening the front door on the system unit and looking through the slot on the interlock door. The lights are read as an eight bit binary number (seven bits for Explorer II) with the lowest amber LED as the least significant bit. The codes shown below are hexadecimal.

2.8.1 Explorer I STBM LED Codes.

Physical locations of LEDs:

(H)(6)(5)(4)¦(3)(2)(1)(0)¦ (Fault)

H	= Halt	amber
6:0	= status indicators	amber
Fault	= Fault indicator	red

81 = Power failure. The processor took the power failure hardware trap.

82 = The processor took the control store parity error trap. This probably means that the processor's WCS is faulty.

83-87 = Should never occur. If they do the processor is probably faulty.

88 = The processor received an unexpected NuBus error.

89 = Processor failed self test.

8A = No memory. If the processor can find a monitor it will also display ERROR: 00000004 as described above.

8B = No boot device. This crash will only occur if the

processor cannot find a boot device and can not find a monitor on which to display a message.

8C = Microload problems. This will only occur if the processor cannot find a monitor on which to display the message "BAD MICROLOAD FORMAT"

8D = DEI PROBLEMS. This will only occur if the processor cannot find a monitor on which to display the device errors 10 - 13 described above.

8E = Monitor device driver problems. The processor got a non-zero completion code on a call to the monitor device driver.

2.8.2 Explorer II STBM LED Codes.

Physical locations of LEDs:

(H)(B)(L)(R)|(F)(6)(5)(4)|(3)(2)(1)(0)| (Fault)

H	= Halt	amber
B	= Busy	amber
L	= cache hit left	amber
R	= cache hit right	amber
F	= cache filling	amber
6:0	= status indicators	amber
Fault	= Fault indicator	red

with Fault LED on:

selftests:

7F = Processor unable to load code from EPROM

01-05 = Part 1 self-tests (Kernel tests)

06 = Passed kernel selftests, attempting to load remainder of tests and STBM (Part 2)

07-39 = Part 2 Lisp Chip and processor board selftests

3A-3C = Floating Point Board tests

crash codes:

71 = No online devices from which to download

72 = Bad microcode format found during attempted download

73 = Device error during attempted download

74 = No good system memory found

75 = NuBus error during download from Nubus memory to internal memories

76 = MCR partition requires floating point board which is not present

Fault LED off:

0n = STBM arbitration phase, looking at slot n

1n = NVRAM search phase, looking at slot n

2n = Monitor search phase, looking at slot n

3n = Memory search phase, looking at slot n

4x = STBM testing chassis slot

41 = ROM test (C3, format version, CRC)
42 = Selftest
44 = NuBus test
48 = Interface diagnostic

5n = Keyboard search phase, looking at slot n

60 = At top level STBM menu

61 = Attempting default boot

62 = Building device menu

63 = Waiting for load device to come ready

64 = Reading partition from load source

65 = Processing MCR sections (except last)

66 = Loading WCS, PDL to A/M, and enter new code

70 = Waiting for first Secondary event

71 = Processing first Secondary event

72 = Waiting for second Secondary event

73 = Booting quietly

78 = Waiting RAM download (P3 mode 5)

2.9 MICROLOADS

The writable control store and other internal memories of the Explorer processor are loaded from a microload. A microload is read by the boot PROM from a mass storage device, interpreted, and the internal memories are loaded.

The Explorer microassembler produces an output file in the "MCR" format. The file name will be "xxx.mcr" where "xxx.lisp" is name of the source file. The "load-mcr-file" function converts the "mcr" file to the microload format and installs it as an "MCR" partition on a disk. This compact representation can be loaded by the Device Driver on the disk controller board when directed to do so by the processors bootstrap PROM. This format provides for the loading of I-mem, A/M memories, D-mem, and main memory. On the Explorer II it also provides load data and initialization instructions for additional onboard IO space structures. The Explorer II MCR format is specified in Appendix A.

2.10 INTERFACE BETWEEN BOOT PROM AND MICROLOADS

When a microload is loaded by the boot ROMs, certain information is passed to the microload in dedicated A memory locations. The following A memory locations are used for the purpose of passing parameters between the boot ROM and microloads:

variable name	A memory location
A-BOOT-COMMAND-BLOCK	#x3F8
A-BOOT-LOD-DEVICE	#x3F9
A-BOOT-MEMORY	#X3FA
A-BOOT-MONITOR	#x3FB
A-BOOT-KEYBOARD	#x3FC
A-BOOT-DEVICE	#x3FD
A-BOOT-MCR-NAME	#x3FE
A-BOOT-LOD-NAME	#x3FF

The boot ROM sets up the following locations as parameters passed to a microload that is loaded:

A-BOOT-MEMORY is set to Fs000000 where s is the slot number of either the first memory board found or (if in secondary mode) the memory specified for this processor in the Command Block received from the STBM Primitive.

A-BOOT-MONITOR is set to designate the system monitor. The slot number is in the most significant byte and the unit number is in the three least significant bytes.

A-BOOT-KEYBOARD is set to designate the system keyboard. The format is the same as for a-boot-monitor.

A-BOOT-DEVICE is set to designate the boot device. The format is the same as for a-boot-monitor.

A-BOOT-MCR-NAME contains the name of the microload in ASCII little endian format.

A-BOOT-LOD-NAME contains the name of the load band. The format is ASCII little endian. If a load band was not selected, this word will contain a value of binary zero.

The boot ROM will only set up A-BOOT-COMMAND-BLOCK if the Explorer processor is being booted as a secondary processor, in which case it shall have the NuBus address of the Command Block.

If Menuboot is run, it will store the system load name in A-BOOT-LOD-NAME in ASCII little endian format. If the system load is on a different unit than the microload, then A-BOOT-LOD-DEVICE will be set to that unit. Otherwise, it will contain the same value as A-BOOT-DEVICE. If Menuboot is not run, A-BOOT-LOD-DEVICE will contain a value of binary zero. The system microcode must check this variable to determine where to get the system load.

In order for Menuboot (or other WCS microcode programs) to request the load of another microload into WCS, there are special processor dependent mechanisms that must be used to reenter ROM. The following A memory locations must be set up for all processors: A-BOOT-MEMORY, A-BOOT-MONITOR, A-BOOT-KEYBOARD, A-BOOT-DEVICE, A-BOOT-MCR-NAME. For the Explorer I the requesting code must turn off the PROM-disable and Bus-Error-Trap-Enable bits in the Machine Control Register (MCR) and then jump to PROM location #x1E. For the Explorer II the requesting code must enable refresh, IROM, and memory cycles in the Machine Control Register (MCR), set the highest M-memory register to #xC3, and then jump to IROM address at #xFF.

The Explorer II ROM code also supplies information to the downloaded code about the type of load which has just occured.

PDL location 0 is loaded with a value indicating the boot type:

0 = STBM, default mode
1 = STBM, non-default
2 = Secondary, default
3 = Secondary, non-default
4 = Special RAM load mode

2.11 CONFIGURATION PARTITION, PRIM ALGORITHM, AND DOWNLOADING

A Configuration Partition is a 17 block disk partition which supplies boot time parameters for a system. These include information such as which software shall be loaded by each downloadable processor and controller and values which allocate ownership of system resources amongst processors. The partition is divided into two parts: the partition header and the configuration modules.

1. The partition header is one block long and is divided into two parts which are both a half block long. The first half of the headers contains the following:

   a. At offset 0 the four characters "CNFG" are found. These characters are used to validate the fact that this is a configuration partition.

   b. The next two characters are used as a CRC value. The "primitive" does not use this.

   c. The next four characters are used as the generation and revision values. The "primitive" does not use these values.

   d. The remaining portion of the first half of the configuration partition is available for comment space.

2. The entries in the second half of the header are called pointers. Pointers are 16 bytes long. The following byte values are relative to offset X200 of the header block. The pointer values are:

   a. Bytes 0,1: A pointer to a configuration module. The pointer is relative to the start of the configuration partition. If the pointer value is 0, this implies that this entry is empty.

   b. Bytes 2,3: Length in blocks of the configuration partition.

   c. Bytes 4 - 7: A boot timeout. This is primarily for multiprocessing systems and is not used by

the Explorer.

d. Bytes 8 - B: The count of configuration entries. This value will be discussed in the configuration module portion of this section.

e. Bytes C,D: A CRC value for the configuration module associated with this entry.

f. Bytes E,F: The board type. If the board associated with this entry is a processor, then the value in this position is equal to the processor value found in the processor configuration ROM at location FSFFFF9C. For a controller, this value is equal to the disk partition type assigned to the download software for a particular controller.

g. Bytes 10,11: Implies via a 1 in a bit position, what slot the board may be in. For instance, an Explorer I board can only be in slot six so bit six would be set to a one for the Explorer I board. Some boards can be in any slot so this value would be all ones for those boards. Notice a configuration partition assumes a sixteen slot chassis.

h. Bytes 12,13: Indicates board type. The value 1 indicates that this entry expected a processor board. A 2 indicates that this entry expected a downloadable controller. No other values have meaning.

i. Bytes 14 - 1F: Reserved.

3. The next entries in the configuration partition are called configuration modules. They can be accessed by another processor when they are in main memory. This feature is not used by the Explorer processor. However, much of the information in the configuration module is used by the Explorer "primitive". The following byte values are relative to the beginning of the configuration module. The information in the configuration module is as follows:

a. Bytes 0 - 3: The busy status flag. This flag is used when the configuration is in memory. The Explorer "primitive" does not use this field.

b. Bytes 4 - 7: NuBus memory base address. This is not used by the Explorer "primitive".

c. Bytes 8 - B: Monitor slot and unit. This value

is set up by the Explorer Primitive.

d. Bytes C - F: Keyboard slot and unit number. This value is set up by the Explorer "primitive".

e. Bytes 10 - 13: Disk slot and unit number where the Lisp MCR partition is to be found. The Explorer "primitive" expects this value to already be set up. If a value of all ones(-1) is found, a default disk is assumed. The assumed default disk is the disk from which the Explorer "primitive" was loaded. If bytes 12,13 of the pointer entry is a 2, then this entry is the disk address of a download software partition.

f. Bytes 14 - 17: A four-character ASCII name of either the Lisp MCR partition or the download software partition. Bytes 12,13 of the pointer entry pointing to this configuration module indicates which one of the two this is. Note, all four characters must be exactly as found on disk. A name with less than four characters must be blanked filled.

g. Bytes 18 - 1B: The disk slot and unit number of the configuration partition currently being used by the Explorer "primitive". This value is set up by the Explorer "primitive".

h. Bytes 1C - 1F: The four character ASCII name of the configuration partition being used by the Explorer "primitive". This is set up by the Explorer "primitive".

i. Bytes 20 - 23: The synchronization flag. This is for multiple processors and not used by the Explorer "primitive".

j. Bytes 24 - 43: The 32 character hardware identification value. This value is used only when downloading is required (bytes 12,13 of the pointer field = 2). This is a part number followed by a three character ID. This value must match exactly the information found in the configuration ROM on the board to be downloaded or no download occurs.

k. Bytes 44 - 63: A list of 32 byte ASCII entry. The number of entries in the list is in bytes 8 - B of the pointer entry for this configuration module. Note, these entries are predefined and are strictly formatted. A removal or addition of a blank within these ASCII entries will cause the

"primitive" to fail. For an Explorer the entries are as follows:

- Entry 0: The part number and id value as described above.

- Entry 1: ASCII value "Explorer Processor" or "Explorer II Processor". The Explorer primitive keys off the string "Explorer".

- Entry 2: "Slots owned: (ASCII slot numbers each separated by a space). This entry is not used by the Explorer primitive.

- Entry 3: "Load Slot :(Single digit ASCII slot number of the disk unit containing the Lisp load band)".

- Entry 4: "Load Unit :(Six digit ASCII unit number of the disk containing the Lisp load band)". Note, if either entry 3 or 4 is an asterisk, then a default disk is used to load the Lisp load band. The disk defaulted to is the same disk from which the primitive was loaded.

- Entry 5: "Load Name :(name of the Lisp load band)". If an asterisk is used, then the first Lisp load band with the default bit set in the disk partition table is used. The primitive searches for a partition of type Explorer or of type TI Lisp.

2.11.1 The Algorithm.

The following describes the algorithm in determining if a board in the system matches and entry in the configuration partition. If a match is made then either a download occurs or a CPU entry is pushed on the stack.

First, the primitive scans each NuBus slot searching for board in a NuBus slot begining with slot 0. If a board is found a search is made through the pointer entries in the configuration

partition to find a 1 bit in the corresponding bit position in bytes 10,11 of the pointer entry. If a match is not found, the search for a board in higher slot numbers continues. If a match is found, the primitive fetches bytes 12,13 of the matching pointer. If the board found is not the same type as specified in bytes 12,13 of the pointer the above search continues. The "Same Type" check fetches byte 12,13 of the pointer to determine if the pointer entry expected a controller board or a CPU in this slot. If the board in the slot was a controller and the pointer entry of the configuration partition expected a a controller board, the the controller type is checked. If the types compare then a download is performed assuming there is a download software partition for this board. If the board type was a CPU and the pointer expected a CPU, the the configuration module address for this CPU is pushed onto a stack for later use; this assumes the CPU types matched. If no match occurs, then the slot search continues. The algorithm is complete when all slots have been examined.

2.11.2 Downloading.

Downloading is a method to patch ROM code on a controller board by inserting new code into the RAM portion of the board and using the RAM resident code rather than the ROM resident code. Previous discussions indicated that the "primitive" performed the downloading. This is only partially correct. Once the primitive has determined that the a board is to be downloaded, the disk is searched for the software partition associated with that board. The software partition contains Diagnostic Engine Code which has been previously discussed in this section. The Diagnostic Engine code performs the actual downloading. The software partition found by the primitive contains the Diagnostic Engine code in the first portion and the ROM replacement code in the second portion of the partition.

Once the Diagnostic Engine Code partition has been loaded by the primitive, it is then interpreted by the primitive's DE code interpretor. At the completion of the DE code's execution, an error indicator is checked by the primitive. If an error occurred during downloading, an error message is generated. Otherwise the algorithm continues.