XEROX

PARC

27 January 1983

To: Dolphin Users and Maintenance Personnel

From: Edward Fiala

Subject: Dolphin Booting and Maintenance Panel Codes

Filed On: [Indigo]<D0Docs>MPCodes.Bravo, .Press

This memo revises the 10 October 1980 memo by Hal Murray about maintenance panel codes; it is incomplete, possibly inaccurate for TOR configurations. Please report any discrepancy or omission. It intends to provide enough information to diagnose malfunctions to the extent that MP codes permit, covering both Pilot and Alto microcode systems. Lisp and Smalltalk presently use the same MP codes as Alto, and Cedar the same as Pilot.

Please note that MP codes produced by Initial (one of the microprograms discussed below) during booting changed about 1 June 1982 and about 10 January 1983. If you are using a pre-Trinity version of Pilot or Alto emulator microcode produced before 1 June 1982, then this document will be inaccurate with respect to Initial. If you Etherboot Initial (which happens if you have just powered up, and your SA4000 disk is not yet up to speed), then this memo should be accurate.

Some MP codes shown by Pilot software are given, but this is NOT authoritative with regard to MP codes generated by software--only the ones from the microcode are reliable.

Some recent MP code changes are as follows:

1) MP codes reported on faults are now the same for Initial, Pilot, and Alto microcode. These are enumerated at the end of this document. "Faults" are abnormal events such as parity errors, stack overflows and underflows, or illegal map or write protect violations--normally a fault represents a hardware failure, and it may occur at any time.

2) Improvements have been made to map and storage tests in Initial; imperfect storage boards are now reported in the MP, even when initialization completes in spite of them. Because of improved testing, some storage experiencing failures will not be used by the new Initial, even though that storage was acceptable to the old Initial. Dolphin maintainers and users may be able to use the new MP codes to immediately identify failing storage boards. The Initial released 10 January 1983 shows in the MP not only the failing storage board but also the Blk.0, Blk.1 and Syndrome values which uniquely identify a failing RAM.

3) The new Initial does not distinguish configurations with CSL microswitch keyboards from those with Star keyboards. Since both Pilot and Alto emulators now handle both kinds of keyboards (using microcode overlays), this distinction is no longer useful. As a result, boot servers have replaced AltoLF.Eb and AltoCSL.Eb by a single file AltoD0.Eb. Tor configurations, which use a UIB terminal interface rather than a UTVFC interface, are still distinguished, however.

4) On a CSL keyboard, Initial's map and storage tests may be looped by keyboard booting with the "7" key depressed. When you do this, the tests repeat until some failure is detected; then Initial repeatedly sequences through the four MP codes describing the failure.

Booting

A Dolphin bootstrap takes place in five stages, as follows:

(1) Hardware loads the Boot microcode from an EPROM;
(2) **Boot** tests the processor and loads Initial;
(3) **Initial** tests the map and storage and loads an emulator;
(4) **Pilot** or **Alto microcode** initializes itself, then loads and starts the Pilot germ or Alto breath-of-life program;
(5) Alto OS or NetExec or Pilot software is booted.

Also, the RunPilot Alto program used to bootstrap Pilot commences a bootstrap at step (3), and Lisp, Smalltalk, and OthelloD0 (from the Alto MesaNetExec) are bootstrapped starting at step (4). However, I think RunPilot embeds a very old version of Initial which may not report the MP codes given in this document.

A bootstrap may be initiated in the following ways:

(1) by pushing and releasing the start (power-on) button;
(2) from the test hardware (Midas);
(3) from a TOD clock (not standard equipment);
(4) from a watchdog timer (not standard equipment);
(5) by executing the Boot function in a microinstruction;
(6) pushing the keyboard boot button (CSL keyboards only).

In addition to these normal methods of booting, if your machine is sick, it might spontaneously boot itself when a fault happens while the fault task is running. If the problem is really bad, it may reboot over and over.

On machines with CSL keyboards, repeat booting may occur when the display is powered down; the back channel reports continuous null in this case, perceived by microcode as "boot button depressed". This doesn't happen with Star keyboards.

Also, some deficiencies in error handling by the Initial microprogram cause certain classes of recoverable hardware failures (disk and ethernet problems) to reboot the machine rather than recovering from the error.

If your machine won't boot or boots very slowly, it is important to go through the following check list:

1) If you have just powered up, make sure your display is turned on--sometimes the emulator won't run if the display is powered off, and you obviously won't be able to see anything.

2) If you have a CSL keyboard, sometimes the keyboard microcomputer will power on in a bad state; if this happens, you can have all kinds of trash happen on the

backchannel--erroneous keystrokes, mouse button clicks, and mouse movement. To fix this, push the keyboard boot button.

3) Otherwise, watch the MP while the problem is happening. The detailed sequence of numbers may indicate what is going wrong. You will have to get your head down low to observe the numbers on the MP reliably; people have frequently reported numbers to me with 1's translated into 7's, and some other observation errors are occasionally made.

Hardware Boot

While you depress the start button, the hardware shows 8888 in the MP as a light test. When you release the power-on button, the machine will then run through the boot sequence discussed above. During the hardware boot, you might see either 8808 or 8880 in the MP if RM or IMX parity errors are detected by the hardware during loading. Control is transferred to location 0 after loading.

EPROM Boot Microcode

Boot is a tiny diagnostic and bootstrap loader limited (for reasons discussed in the D0 Hardware Manual Section 4.10) to 1024 instructions and 16 immediately accessible RM registers; other registers can be referenced using the stack pointer, however. Its function is to test the parts of the machine needed for the next stage of the boot thoroughly but quickly, reporting any failures on the MP. If these tests are passed, it then boots the Initial microcode from an io device; the SA4000 disk and the 3 mb Ethernet are boot devices; Tor configurations use a variant Initial that boots from floppy disk; and 10 mb Ethernet booting will probably be made available at some point. Because loading Initial from the disk or Ethernet requires most of the processor to work, the tests performed by Boot must include most of the machine. However, the map and storage are not needed during the boot of Initial, so these parts of the machine are not tested. Also, many processor functions needed only by the emulator are not tested by Boot.

Boot runs a few processor tests to find out whether or not the processor is healthy enough to continue loading. Many machines malfunction when first powered up, then work correctly. If the processor tests fail, Boot will show an error MP code (0000 to 0039) for a second or so and then reboot. Otherwise, registers are initialized.

Then Boot determines whether or not it was started from the maintenance panel (i.e., by Midas during debugging). If so, it will show the MP code 0002 and read the Kernel debugging program over the printer interface from Midas.

Otherwise, Boot tries to read the first program on the SA4000 boot record (normally Initial) directly into the microstore. When Boot starts this, it will show 0040 in the MP. If you see this (or 0041 or 0046 which frequently follow immediately), your processor is at least somewhat alive (Initial lives in a special reserved portion of the disk, so you won't see it among your Alto files.).

If the disk won't work (0041 to 0045, 0047 to 0048) or isn't ready yet (0046), Boot will try to obtain Initial.Eb from the Ethernet; when this decision is made, a one second pause allows you

to read the MP; if the reason for the boot is NOT a button push, the delay is extended to one minute to prevent a sick machine from hogging the boot servers. When Ether booting starts, 0060 (trying to Etherboot) will appear in the MP; other 006x MP codes indicate Etherboot problems. While waiting for a boot server to respond and while transmitting Initial microcode from the boot server, 004x slowly alternates with 006x, so that you can see both the reason why the disk boot failed and the current Etherboot indication.

Unfortunately, many machines experience a short period of unreliability after being powered on, but then work correctly. These machines encounter the one minute wait intended to prevent a sick machine from hogging the boot servers, and this can be frustrating, if you are waiting for the machine to become ready. If your machine does this, keep pushing and releasing the start button until you see a healthy 0060.

The standard trick for *forcing* an Etherboot is to turn power off and then back on. It takes the disk about two minutes to become ready again. If you push the start button before the disk is ready, you should get to the Alto NetExec.

Note that Initial is loaded directly into the microstore without using either the map or storage.

Initial Microcode

Initial is primarily responsible for testing and initializing the map and storage, reporting any failures in the MP, then loading and starting an emulator; for Pilot, it also loads the germ. Initial should also test whatever was not tested by Boot, but it doesn't presently do so. Much of the machine is unfortunately not tested at all by any program during the boot sequence.

Initial has a number of entry points which determine the emulator booted later and the io device used for the boot:

SA4000 Pilot boot;
3 mb Ethernet Pilot boot;
SA4000 partition 1 Alto start;
SA4000 partition 2 Alto start;
3 mb Ethernet Alto start.

Other entries are used by test programs.

The above entry point is overruled when you have carried out a keyboard boot (CSL keyboards only) with the "7" key depressed while any emulator was running. In this case, Initial's map and storage tests will repeat until a failure is detected; then the four MP codes describing the failure will be shown in sequence indefinitely.

The 'SA4000 Pilot boot' starting address will boot the Pilot microcode and germ installed on your SA4000. This entry point is used when you install InitialPilot.Eb as your initial microcode and by RunPilot.bcd in your Alto volume; it is also used on a keyboard boot (CSL keyboards only) when you have depressed the "p" key while any emulator was running.

When Boot obtains Initial.Eb from the Ethernet, it starts it at the 'Ether Alto start' entry point, which (later) will cause the Alto emulator and NetExec to be obtained from the Ethernet also. There is currently no way to boot a Pilot program directly from the Ethernet. If your disk is sick and you want OthelloD0, you have to cycle power, if necessary, to force an Etherboot and get

to the Alto NetExec; then ask the NetExec for the MesaNetExec; and, finally, ask the MesaNetExec for OthelloD0. If you are running the Alto OS, you can go more directly to the MesaNetExec with 'Etherboot 100'.

Shortly after Initial receives control, it puts 0700 ('starting map test') into the MP very briefly followed by 0400 ('starting storage test'); if you see 0040 then 0700 or 0400 (without an intervening 0060), your RDC is at least somewhat healthy since Initial was loaded from the disk. Initial first tests the map; it will hang with the 'bad map' MP code (0702) displayed, if the map is imperfect.

Then Initial tests storage and uses only 'good' pages. 0400 is seen in the MP during storage testing, which lasts less than 2 seconds with four perfect 96k-word storage boards.

If Initial detects any storage imperfection, it will do additional testing, and 0400 will be shown for 4 to 9 seconds (timing approximately proportional to the amount of storage). If the number of pages with correctable failures exceeds 1/8 of all pages, and if the amount of good storage is less than 128k words, then the entire test will be repeated allowing the pages with only soft failures to be used; otherwise, only perfect pages are used. After all testing is complete, four numbers will be shown in the MP for about 1.0 seconds each:

1) 0400 + 1 if 1st storage board imperfect,
+ 2 if 2nd imperfect,
+ 4 if 3rd imperfect
+ 8 if 4th imperfect,
+ 16 if 5th imperfect,
+ 32 if 6th imperfect,
+ 64 if 7th imperfect,
+128 if 8th imperfect.

This interpretation is for the 96k-word storage boards with 16k RAMs. Each such board occupies 128k words of real address space but implements only the first 96k words of the space. Hence, the bits actually represent imperfections in the existing storage of each 128k-word bank of storage.

2) Hard bad page count (some uncorrectable failures). This count, and the imperfect board indication previously, are incomplete if a page with hard errors is thought to be a non-existent page. The current algorithm is very unlikely to make this mistake.

3) Soft bad page count (correctable failures only).

4) Blk and Syndrome bits for the last failure detected by the error correction hardware. This is shown as four octal characters BSSS, where the first character of the MP code is the two Blk.0 and Blk.1 bits and the last three characters are the Syndrome.

These numbers are NOT SHOWN when all storage boards are perfect.

Even when some storage is bad, unless the amount of 'good' storage is reduced by failures to less than 64k words, initialization will continue normally following the bad-page MP codes.

When the storage test has finished, consecutive map entries enumerate pages of 'good' storage and storage has been zeroed; map entries above the last good page are initialized to Vacant. The counts of good pages and hard and soft bad pages and the bad-board word are stored in

high RM words where they will not be smashed by Pilot or Alto. A Pilot opcode allows these test results to be accessed and reported.

On an Alto disk boot, Initial then puts 0720 into the MP and continues reading the SA4000 boot record. But this time the emulator from the boot record is placed into storage rather than directly into the microstore. On an ether boot, it instead shows 0758 (Star or CSL), or 0762 (Tor) in the MP and reads Alto.Eb from the ethernet into storage. When it is going to start Pilot, Initial instead shows 0725 in the MP; this kind of boot is more complicated since it has to locate and load the germ as well as the emulator microcode.

At the end of file, the microcode image in storage is loaded and started with LoadRAM.

With the old Initial, normal sequences are as above except that 0760 is shown instead of 0758 for a CSL keyboard on an Etherboot. Also, 0400 is not shown, and the old Initial zeroes and uses any storage that it finds without testing. If it can write and read one value from the first quadword of a 64-quadword page, it assumes the entire page is good and uses it. Also, the old Initial reports different MP codes on faults from those in the table at the back of this document.

Emulator Microcode

Early in initialization both the Pilot and Alto emulators show the MP code 'Start device init' (0104) barely long enough to see. Seeing 0104 means that Initial tested and zeroed storage, loaded the microcode image into storage from the disk or ethernet and then from storage into the microstore, transferred control to it, and executed at least the first few microinstructions successfully.

After initializing device drivers, Pilot then shows the number of pages found to be good by Initial; this will be up long enough to see (~ 0.4 seconds). Pilot microcode initialization then finishes by starting the Germ, which will show 0900 followed by the sequence given in the tables later.

For the 96k-word storage boards using 16k RAMs, the MP will show 96K/256 pages/board which is 384 x number of storage boards (i.e., 0768, 1152, 1536, 1920, 2304, or 2688). Some MPs will show this value plus 1 occasionally for unknown reasons.

The Alto emulator has several different entry points, as discussed below. After 0104, on a disk boot, 0118 (GotBreathOfLife) is shown for 0.3 seconds after the first page from the disk boot record has been read successfully. On an ethernet boot, 0114 (start booting the NetExec) is shown, then 0118 when the breath-of-life has been received from the Ethernet boot server. Finally, on both disk and ethernet bootstraps, the Alto emulator shows the number of good pages found by Initial in the MP. Then it loads the final microcode overlay and starts the emulator at the breath-of-life program's disk boot or Ether boot address.

An emulator is usually started by Initial, but Lisp and Smalltalk are started directly, preserving the Alto PC and disk partition. When Initial doesn't immediately precede an emulator, the number of good pages displayed will be correct unless you have run the Audio or Jasmine microcode since you last booted the hardware--these io drivers smash the results left earlier by Initial.

When started at its normal entry point, the Alto emulator will boot the OS from SA4000 partition 1 only if you have not used the keyboard boot button kludge (which is only available with CSL keyboards, not with Star or Tor terminals).

If you depressed the "0", "p", or backspace key and then pushed the keyboard boot button when your emulator was correctly running the UTVFC task (i.e., refreshing the display), then the OS will be booted from SA4000 partition 2 ("0" typed) or the NetExec from the Ethernet (BS typed), or Pilot will be booted using the Pilot microcode and germ installed on your disk ("p" typed). If none of these keys was depressed, or if some other keys were also depressed, then a partition 1 disk boot occurs.

Pushing the keyboard boot button does nothing if an emulator isn't running.

Microcode Coordination

If you install microcode on your disk, you can be reasonably sure that it won't change out from under you, but if you EtherBoot, you will get whatever microcode is currently installed for your type of machine on the nearby boot servers.

Note that all the boot servers try to automatically keep their boot files up-to-date. This makes it hard to have two different versions of the microcode available in different geographic locations. It is possible to defeat the automatic update heuristics. See Hal Murray if you really need to know more. Currently, things are even more hectic than it appears since the network is partitioned, and some boot files are not automatically updated.

The two microcode systems you might want to install on your disk as *initial microcode* (not including Lisp and Smalltalk) are:

> InitialAltoD0.Eb, the concatenation of Initial.Eb and AltoD0.Eb; it will load the microstore with Alto and Alto-style Mesa emulators and start the Alto emulator at its normal starting address after running Initial. If you use the "p" keyboard boot feature, then a Pilot boot will be carried out using the Pilot microcode and germ installed on your disk.
>
> InitialPilot.Eb, a version of Initial.Eb that will boot Pilot using the pilot microcode and germ installed on the Pilot portion the disk; the pilot microcode can be either PilotD0.Eb or CedarD0.Eb. The Tor system uses a variation of InitialPilot with which I am not very familiar.

During the transition between Rubicon and Trinity versions of Pilot, there will be two versions of each of these programs available. This is necessary because Pilot booting is different for Trinity and Rubicon.

(Details of installing microcode and such are described in the Othello document.)

Normal MP Code Sequences

If your hardware is working properly, the viewable MP sequence if you disk is not yet up to speed is

8888, [46, 60], 700, 400, 758, 104, 114, 118, NPages,

where "46, 60" may repeat several times before continuing with the rest of the sequence, if your boot server is busy; the 758 will be 762 with a Tor UIB terminal. Other MP codes are not up long enough to see unless something goes wrong; 700 is up barely long enough to see.

If you have just powered up your machine, it might get a 22 or 24. I am also suspicious that a microcode bug in the boot microcode may be causing 84 (unexpected wakeup from task 16b). After a second or so, your Dolphin will automatically reboot, but this time 46 will stay in the MP for a minute or so. If you get tired of waiting, poke the button again.

The viewable MP sequence for a "normal" InitialAltoD0.Eb disk boot is

8888, 40, 700, 400, 720, 104, 118, NPages.

Lisp and Smalltalk show the same sequence starting at 104.

If your disk is ready and setup to boot Othello, the following sequence of numbers should appear in the MP when you boot:

8888, 40, 700, 400, 725, 104, NPages, 910, 9x0, 930, 940, 9x0, 990.

The x indicates that it flickers faster than I could read. If you are booting Tajo or Cedar directly, then other numbers may be there. The same sequence, without the leading 8888, will appear on a "p" keyboard boot. The same sequence should occur when you call RunPilot.bcd from your Alto volume, but without either the leading 8888 or 40.

In the following pages, a # indicates a final MP code. The machine will hang with this number in the MP until you boot again. All other MP codes are either errors that will be retried automatically or simple indications of progress.

MP Codes from the hardware

8888	Lamp test (shown while the start button is down).
8808#	RM parity error. The MP freezes with this code, indicating (??) a failure during the hardware boot.
8880#	IMX or control store parity error. The MP freezes with this code indicating (??) a failure of the hardware boot.

MP Codes from Rev L EPROMs

0000	One of the first instructions in the EPROM clears the MP; seeing this number means that the ALU all-zeroes, all-ones, or GoTo tests failed.
0001	The ALU all-zeroes, all-ones, and GoTo tests passed. You should never see this since some other error should happen or the MP should change to 0040 when RDC booting starts.
0002	Midas boot.
0004 to 0015	One of the preliminary ALU tests failed. See Boot.mc for the details. *This MP code list may be buggy in this area*.
0004	XOR failure using 0 constant.
0005	XOR failure using 0 from T.
0006	ALU failure using 125b constant.
0007	ALU failure using 125b from T.
0008	ALU failure using 252b constant.
0009	ALU failure using 252b from T.
0010	ALU failure using 125000b constant.
0011	ALU failure using 125000b from T.
0012	ALU failure using 52400b constant.
0013	ALU failure using 52400b from T.
0014	ALU failure using 177400b constant.
0015	ALU failure using 177400b from T.
0016	Mismatch after write then read of an RM register via the stack.
0017	The contents of an RM register have changed.
0018	Register read and compare error using the Stack.
0020 to 0035	A fault happened. The MP contains 20 plus the contents of the Parity register. A fault in the fault handler will reboot the machine, so you may not get to see these codes. The value shown is 20d + (1 if memory error) + (2 if RM parity error) + (4 if control store parity error) + (8 if stack overflow or underflow).
0020	A Breakpoint microinstruction (i.e., one containing the SetFault function) was executed.
0021	Memory error. Since the EPROM code doesn't touch the map or storage, this is probably an H4 Parity error.
0022	RM register parity error.
0024	Control Store parity error.

0022 and 0024 are to be expected if you have just powered up your machine. (The bias on the RAM chips hasn't been pumped up yet.) This will invoke a one minute delay to avoid hogging the boot servers when something is broken. Poke the button again if you want faster service.

0028	Stack error.
0040	Starting to load microcode from RDC.
0041	Can't find RDC. (Will now try to EtherBoot)
0042	SA4000 disk read error.

0043 SA4000 seek timed out.
0044 SA4000 disk checksum failure.
0045 SA4000 bad Control Store address--attempt to load into EPROM area.
0046 SA4000 disk not ready. (Will now try to EtherBoot)
0047 The label word which should contain a link to the next page of microcode to be loaded has an invalid disk address.

Most disk errors (0042, 0043, 0044, 0045, 0047) can be caused by simple transient read problems. The RDC task simply retries all of them while the emulator task is counting down a timer. If the timer runs out, you will see 0048.

0048 Didn't load microcode from RDC within 1 second. (Will now try to EtherBoot)
0060 Trying to load microcode via Ethernet.
0061# Can't find Ethernet board.
0062 Bad Ethernet checksum while reading microcode.
0063 Bad Control Store address--attempt to load into EPROM area.
0064 Hardware error (bad status) at end of packet.
0065# Timeout after 15 tries at about 10 seconds each.

If EtherBooting doesn't work, the MP will slowly alternate between 006x and 004x so that you can see both what was wrong with the disk and what is wrong with the Ethernet. If the Etherboot eventually times out, you will see 0065 alternating with the bad disk code.

0070 to 0085 An unnexpected wakeup happened. The MP contains 70 plus the number of the offending task.
0071 Unexpected wakeup for task 1.
0072 Unexpected wakeup for task 2.
0073 Unexpected wakeup for task 3.
0074 Unexpected wakeup for task 4. (RDC or Ethernet output.)
0075 Unexpected wakeup for task 5.
0076 Unexpected wakeup for task 6.
0077 Unexpected wakeup for task 7.
0078 Unexpected wakeup for task 10B. (Ethernet input.)
0079 Unexpected wakeup for task 11B.
0080 Unexpected wakeup for task 12B.
0081 Unexpected wakeup for task 13B.
0082 Unexpected wakeup for task 14B.
0083 Unexpected wakeup for task 15B.
0084 Unexpected wakeup for task 16B (timer task).
0085 Unexpected wakeup for task 17B (fault task).

If you get one of these, one of the IO controllers is probably broken. For example, its reset logic is not working, or the wakeup logic on the RDC or Ethernet board is generating the wrong task number.

0104 Loading SCB Initial from floppy disk--don't confuse this MP code with StartDeviceInit (also 0104) from emulators; the preceding sequence of MP codes disambiguates them.

MP Codes from Initial

Fault handler MP codes in the range 100 to 255 may also happen when running the new Initial; these are in a different table because they are common to Initial, Pilot, and Alto.

0700	Start map test.
0702#	Bad map.
0400	Start storage test.
0401 to 0655	Failures detected on one or more boards. MP shows for about 1.0 seconds: 0400 + 1 if storage board 1 had failures + 2 if board 2 + 4 if board 3 + 8 if board 4 + 16 if board 5 + 32 if board 6 + 64 if board 7 +128 if board 8. The above is the interpretation for the 96k-word storage boards; +1 actually means 'the first 128k words'; +2 'the second 128k words'; etc.
NHardBad	Count of hard bad pages (ones with uncorrectable failures), where pages are 256 16-bit words; shown after 0401-0655.
NSoftBad	Count of soft bad pages (only correctable failures); shown after NHardBad.
BlkSyndrome	A four character octal number in which the first character is the value of Blk[0:1] and the last three characters are the Syndrome of the last storage failure detected by the error correction hardware; shown after NSoftBad. A translation table from values of BlkSyndrome to chip positions on the storage boards is given in a table later.
0701#	Not enough memory. (Initial requires 64K words.)
0720	Starting to load emulator microcode from disk.
0721#	No RDC.
0722#	RDC Read error.
0725	Starting to boot Pilot from disk.

0730 to 0736 appear during floppy disk booting which only happens on Tor and ESS configurations, which have different Initial microcode.

0730#	Start floppy disk loader.
0731#	Mesa never ready.
0732#	Illegal for floppy disk booting.
0733#	Bad floppy disk status.
0734#	Execute input channel transfer not found.
0735#	Loader ended.
0736#	Invalid FIFO type.
0741#	Can't find UTVFC (or Tor display controller).

*When Initial starts Etherbooting, it puts 740+2*bfi into the MP. (bfi is related to the boot file number). If EtherLoad can't load that file within a reasonable length of time it will give up and bump the MP by one.*

0758	Trying to load AltoD0.Eb from the Ethernet.
0759#	Timeout trying to load AltoD0.Eb
0762	Trying to load AltoTOR.Eb from the Ethernet.
0763#	Timeout trying to load AltoTOR.Eb
0774	Trying to load PilotD0.Eb from the Ethernet.
0775#	Timeout trying to load PilotD0.Eb.
0778	Trying to load PilotTOR.Eb from the Ethernet.
0779#	Timeout trying to load PilotTOR.Eb.
0809	Pilot Microcode not installed on SA4000. (Try to EtherBoot.)
0810	Germ not installed on SA4000. (Try to EtherBoot.)
0811	Physical Boot Volume not set on SA4000. (Try to EtherBoot.)
0812	Label check from SA4000. (Try to EtherBoot.)

Syndrome Translation Table for 256k Storage Boards using 64k RAMs

Initial records Blk and Syndrome for the last error correction fault of its storage test and shows these as a single octal number with Blk.0 and Blk.1 displayed in the first character and Syndrome in the last three characters. If the Syndrome is not in the table below, then it is not a single error, so no single chip can be identified. Blk.0 and Blk.1 are irrelevant for the 256k storage boards.

Syn-drome (octal)	Bad Bit 0-63d) (c.0 to c.7)	Location	Syn drome (octal)	Bad Bit 0-63d (c.0 to c.7)	Location
1	c.7	U97	172	30	U70
2	c.6	U107	174	46	U60
4	c.5	U117	177	62	U51
7	0	U155	200	c.0	U160
10	c.4	U127	206	1	U111
13	16	U148	212	17	U101
15	32	U138	214	33	U91
16	48	U121	217	49	U81
20	c.3	U126	227	9	U158
26	8	U84	233	25	U151
32	24	U67	235	41	U141
34	40	U57	236	57	U125
37	56	U48	247	5	U92
40	c.2	U143	253	21	U66
46	4	U113	255	37	U56
52	20	U122	256	53	U47
54	36	U112	266	13	U116
57	52	U102	272	29	U106
67	12	U159	274	45	U96
73	28	U152	277	61	U86
75	44	U142	307	3	U156
76	60	U132	313	19	U149
100	c.1	U153	315	35	U139
106	2	U82	316	51	U123
112	18	U68	326	11	U95
114	34	U58	332	27	U69
117	50	U49	334	43	U59
127	10	U124	337	59	U50
133	26	U114	346	7	U93
135	42	U115	352	23	U83
136	58	U105	354	39	U104
147	6	U157	357	55	U94
153	22	U150	367	15	U87
155	38	U140	373	31	U71
156	54	U103	375	47	U61
166	14	U85	376	63	U52

Syndrome Translation Table for 96k Storage Boards using 16k RAMs

Initial records Blk and Syndrome for the last error correction fault of its storage test and shows these as a single octal number with Blk.0 and Blk.1 displayed in the first character and Syndrome in the last three characters. If the Syndrome is not in the table below, then it is not a single error, so no single chip can be identified.

Syn-drome (octal)	Bad Bit 0-63d) (c.0 to c.7)	Blk =0	Blk =1	Blk =2
1	c.7	U082	U143	U143
2	c.6	U104	U165	U165
4	c.5	U103	U164	U164
7	0	U003	U003	U065
10	c.4	U081	U142	U142
13	16	U005	U005	U067
15	32	U065	U126	U126
16	48	U067	U128	U128
20	c.3	U020	U020	U082
26	8	U011	U011	U073
32	24	U013	U013	U075
34	40	U073	U134	U134
37	56	U075	U136	U136
40	c.2	U042	U042	U104
46	4	U007	U007	U069
52	20	U009	U009	U071
54	36	U069	U130	U130
57	52	U071	U132	U132
67	12	U015	U015	U077
73	28	U017	U017	U079
75	44	U077	U138	U138
76	60	U079	U140	U140
100	c.1	U041	U041	U103
106	2	U026	U026	U088
112	18	U028	U028	U090
114	34	U088	U149	U149
117	50	U090	U151	U151
127	10	U034	U034	U096
133	26	U036	U036	U098
135	42	U096	U157	U157
136	58	U098	U159	U159
147	6	U030	U030	U092
153	22	U032	U032	U094
155	38	U092	U153	U153
156	54	U094	U155	U155

Syn-drome (octal)	Bad Bit 0-63d) (c.0 to c.7)	Blk =0	Blk =1	Blk =2
166	14	U038	U038	U100
172	30	U040	U040	U102
174	46	U100	U161	U161
177	62	U102	U163	U163
200	c.0	U019	U019	U081
206	1	U025	U025	U087
212	17	U027	U027	U089
214	33	U087	U148	U148
217	49	U089	U150	U150
227	9	U033	U033	U095
233	25	U035	U035	U097
235	41	U095	U156	U156
236	57	U097	U158	U158
247	5	U029	U029	U091
253	21	U031	U031	U093
255	37	U091	U152	U152
256	53	U093	U154	U154
266	13	U037	U037	U099
272	29	U039	U039	U101
274	45	U099	U160	U160
277	61	U101	U162	U162
307	3	U004	U004	U066
313	19	U006	U006	U068
315	35	U066	U127	U127
316	51	U068	U129	U129
326	11	U012	U012	U074
332	27	U014	U014	U076
334	43	U074	U135	U135
337	59	U076	U137	U137
346	7	U008	U008	U070
352	23	U010	U010	U072
354	39	U070	U131	U131
357	55	U072	U133	U133
367	15	U016	U016	U078
373	31	U018	U018	U080
375	47	U078	U139	U139
376	63	U080	U141	U141

MP Codes from Cedar and Trinity Pilot Microcode

Fault handler MP codes in the range 100 to 255 may also happen when running Pilot; these are in a different table because they are common to Initial, Pilot, and Alto.

0104	Start device initialization.
NPages	Number of 'good' pages found by Initial, shown briefly after initializing devices and just before loading the final microcode overlay. *This number will be garbage unless you ran the new Initial the last time you booted*, and it will be garbage if you ran Jasmine or Audio microcode since last booting.

MP Codes From Alto Emulator Microcode

Fault handler MP codes in the range 100 to 255 may also happen when running Alto; these are in a different table because they are common to Initial, Pilot, and Alto.

Assembly switches allow the Alto emulator to be assembled for direct startup from Midas with no microcode overlays and no requirement for running Initial first. The 0100 MP code only happens in this debugging configuration, which must do its own map and storage test and initialization.

0100	Start map and storage test and initialization.
0101#	Not enough memory. (You need 512 pages or 128K words--more than the 64k words required by Initial.)
0104	Start device initialization.
0105	Started UTVFC initialization (invisible).
0106	Finished loaded or flushing CSL keyboard overlay (invisible--*only happens on systems with CSL keyboard overlays; debugging systems and some Lisp and Smalltalk systems don't have such an overlay*).
0107	Finished display initialization (invisible--*this will be 0106 with no CSL keyboard overlay*).
0110	Started disk boot (invisible).
0111#	Timeout waiting for disk status.
0112#	Hardware error reading disk (after 10 retries).
0114	Start booting the NetExec.
0118	Breath-of-life read successfully from disk or Ethernet.
NPages	The number of 'good' 256-word pages determined by Initial is put in the MP just before the final microcode overlay overwrites initialization; it normally remains in the MP until you boot or crash. The Alto emulator starts immediately after loading the final overlay.

MP Codes from Pilot and Cedar Software

Fault handler MP codes in the range 100 to 255 may also happen when running Pilot or Cedar; these are in a different table because they are common to Initial, Pilot, and Alto. I think these MP codes are the same for both Rubicon and Trinity Pilot releases.

0800	Not Cedar microcode.
0801	Ancient microcode (release date from VERSION opcode not recent enough).
0802	Incompatible version number.
0900	Germ started loading boot file.
0901#	Germ allocation trap--out of frames.
0902#	Control fault. (Probably an unexpected KFCB or trap.)
0903#	Start trap--attempt to start an already started module (Pilot bug).
0904#	Germ page or write protect fault (Pilot bug).
0909#	Germ unnamed ERROR. (As in ENDCASE => ERROR; Pilot bug)
0910	Germ running doing inLoad or outLoad.
0911#	Germ and physical volume have incompatible version numbers.
0912#	Germ and boot file have incompatible version numbers.
0913#	No physical boot file installed.
0915#	Running TeleDebug server (frequently means couldn't get to CoPilot or no CoPilot on disk).
0919	Germ has returned to caller who has hung.
0920	Germ disk or Ethernet driver running; might mean that no gateway is responding to your machine's call for help in booting. Check the gateway situation on your Ethernet connection.
0921#	Hard device error on device being booted.
0922#	Timeout of boot.
0923#	Broken link in chained boot file. (Something about your germ or boot files is wrong--try refetching and reinstalling germ and boot files.)
0924#	No response from any boot server to Germ's request for an Ether boot file.
0925#	Germ bad packet--unexpected packet sequence number or size.
0930	Initializing Pilot Control and Mesa Runtime.
0931#	Pilot and StartPilot have incompatible version numbers.
0932#	Trap before trap handler ready (Pilot bug).
0935#	Can't get to the debugger or too early in startup to find debugger. Try using Othello's Set Debugger Pointers command.
0936#	Swap to debugger canceled because of H key switch.
0937	Attempting to set the time via the Ethernet; sometimes get this MP code when you are trying to go to CoPilot but failed to boot CoPilot before booting your Client volume.
0938#	Running cleanup procedures, normally before a world swap.
0939#	System.PowerOff called, and no power control relay.
0940	Initializing Pilot Storage.
0947	Drive not ready. (Waiting for it to become ready.)
0950	Scavenging (wait awhile).
0960	Deleting temporary files from previous run.
0970	Spaces being created for non-bootloaded code and data.

0975	Transaction crash recovery: transaction log being processed.
0980	Initializing Communication.
0981	Trying to find a Pup/EthernetOne 8 bit address.
0990	PilotClient.Run has been called.

0990 is what you see after Pilot has finished initialization.

Fault MP Codes from Initial, Cedar, Trinity Pilot, and Alto

The old Initial has different fault MP codes.

115 Unexpected Ethernet output task wakeup (Alto only).

116 No state vector error indicating process code inconsistency (Pilot only); rumor is that this happens when a process that shouldn't page fault does.

117 Two MC2 errors; both MC2A and MC2B pipes indicate errors; since LogSE is illegal, this might mean two consecutive references experienced uncorrectable storage failures. Alternatively, this been caused by violating the Output-Output-PStore4 Gotcha (microcode bug).

119 Stack overflow or underflow. For Pilot and Cedar, this can only happen during microcode initialization because stack errors are subsequently passed through to software. For the Alto emulator, this always crashes.

0120 to 0135 RM or CS parity error, possibly in combination with other errors. MP code is 120 plus:
1 if MC1 or MC2 error;
2 if RM parity error;
4 if CS parity error;
8 if stack overflow or underflow.

Many codes show a multiple of 20d + a task number, where the task assignments are given below. The 'pipe task' is the one issuing the reference causing an error--this can determined from the error pipe; the 'current task' is the one running when the fault aborted execution.

Alto task assignments:		Pilot task assignments	
0	emulator	0	emulator
1	unused	1	unused
2	unused	2	unused
3	unused	3	unused
4	unused	4	1st MIOC or 2nd 3mb Ethernet output
5	color display	5	2nd 3mb input or 3rd 10 mb Ethernet
6	3 mb Ethernet output	6	color display, 2nd MIOC, or 2nd 10mb Ethernet
7	3 mb Ethernet input	7	1st 10mb Ethernet
8	RDC (SA4000 controller)	8	1st 3mb Ethernet output
9	unused	9	1st 3mb Ethernet input
10	UTVFC (display controller)	10	RDC (SA4000 controller)
11	unused	11	unused
12	unused	12	UTVFC or UIB terminal controller
13	unused	13	unused
14	timers	14	timers
15	fault task	15	fault task

The following codes imply no RM or CS parity error.

0136 MC12 error occurred but none of the reasons for it is indicated; i.e., neither

H4PE, MOB, MC1A, MC1B, MC2A, nor MC2B errors are true. Conceivably, this can be caused by an MC1 fault on the reference following a PFetch4, if the PFetch4 experiences error correction. Due to a hardware bug, the fault isn't started soon enough in this case, so an extra non-fault-task instruction is executed. If the extra instruction is a reference it wipes out the MC1ErA and MC1ErB indicators.

It is better to show the pipe task rather than the current task for H4PE and MOB errors, but for H4PE's the hardware doesn't indicate whether pipe A or pipe B was involved, and I couldn't find out whether or not the pipe is indicated correctly for MOB's.

140 to 155 — Map out of bounds indicating virtual address greater than 22d bits. Code shown is 140 + current task. An MOB crash can't happen for Pilot because MOB errors fault to software.

160 to 175 — H4 parity error indicating bad parity on the processor bus used by Input, IOStore4, and IOStore16 references. Code shown is 160 + current task. This can never happen at present since these errors are ignored for all tasks by all microcode systems. However, Initial has an entry point which causes any emulator to treat H4PE's as a crash condition, and if this entry point were used, these errors would crash.

180 to 195 — Some fault when the preceding instruction contains a LoadPage and the fault handler decides to continue execution. This indicates a microcode bug and should be reported. For Pilot, this cannot happen for the emulator; for Alto it is never supposed to happen.

200 to 215 — MC2 crash, indicating correctable storage failure of PFetch1, 2, or 4 with LogSE true in the map entry or an uncorrectable storage failure on any reference. The code shown is 200 plus the pipe task. Since LogSE is presently illegal, this code should indicate an uncorrectable storage failure; some microcode bugs may cause it (e.g., Output-Output-PStore4 Gotcha).

220 to 235 — MC1 crash, indicating a page or write protect violation. The code shown is 220 plus the pipe task. This can't happen for Pilot because MC1 faults are passed to software (except when executing the LoadRAM-and-jump opcode, when they crash); MC1 faults always crash for Alto.

240 to 255 — SetFault or Breakpoint crash. The code shown is 240 plus the current task. This is used by the microcode in a few places when impossible conditions are detected; for unused tasks, it represents an unexpected wakeup.