alpha micro
AMSD journal
ALPHA MICRO SERVICE DIVISION
July 1987
Volume 9, No. 7

In this issue...

Special Articles

Software

General Information

July 1987 **AMSD Journal** Volume 9, Number 7

Editor........ Robin Soto
Assistant Editor........ Sandra Tomlin
Artwork Director........ Bill Briscoe

Contributors to this issue: Robert Currier, Ken Ellson, Ray Ezell, Jeff Kreider, Robin Soto, Sandra Tomlin.

The *AMSD Journal* is published monthly by Alpha Micro Technical Publications for the Alpha Micro Service Division. Please address all correspondence to AMSD Journal, Alpha Microsystems, 3501 Sunflower, Santa Ana, California, 92704.

Subscription Information: Subscription rates are $40 per year. Back issue sets are also available for $150 per set. Each Alpha Micro dealer receives one permanent subscription to the *AMSD Journal* free. For additional subscriptions, send your name, company name, address, customer number (if applicable) with payment of $40 for each subscription to:

AMSD Journal
Subscription Department
Alpha Microsystems
P.O. Box 25059
Santa Ana, CA 92799

AMSD Telephone Numbers

Customer Service	(714) 641-7602
Customer Education	(714) 641-6330
Technical Support	(714) 641-7608
Main Switchboard	(714) 957-8500
Spare Parts/Logistics	(714) 641-6339
AMTEC (1200 Baud Bell 212A)	(714) 545-2290
AMTEC (300 Baud Bell 103)	(714) 545-8546

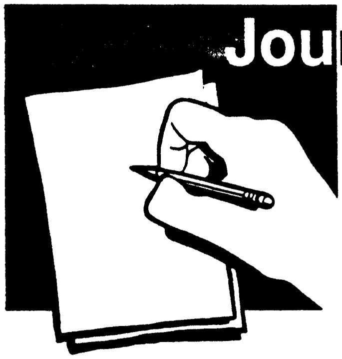
Article Filing Instructions

We hope you find the *AMSD Journal* to be a valuable reference tool, and that you will want to refer to its articles frequently in the future. To make it easy and quick to find information, current articles are designed to be filed with articles from past issues. The entire set of *Journal* back issues forms three volumes: "General Information," "Software Information," and "Hardware Information." (The set of back issues is available for purchase. See "Subscription Information," above.)

The title of each feature article in this issue includes a reference number. Use the reference number to file the article in the back issue volume indicated at the top of each page of the article. For example, if the top of the first page of the article "6.5.5 One Hundred New Uses for MULTI," contains the words "Software Information," you know that article is to be filed in Section 6 of the "Software Information" back issue volume after article number 6.5.4.

The last pages of the *Journal* are new Tables of Contents for the back issue volumes, updated with entries for articles included in this month's issue.

Journal Notes



This month you will see a departure from the usual Journal format-- we feature two articles that focus on different aspects of the same topic: troubleshooting MONTST problems.

We offer this information as two separate articles because we think the information may be of interest to different types of readers.

If you have been puzzled by occasional failures of MONTST to reboot the computer, see Software Volume article 3.3.17 - "Introduction to MONTST Problems" for background information on what factors affect the operation of the MONTST program. This article contains important information on why MONTST can fail, how to troubleshoot MONTST problems, and some hints on how to avoid MONTST problems in the first place.

If you have a specific problem with using MONTST on a computer with an AM-350 board installed and you already feel confident that you are well versed on MONTST's operating characteristics and interactions with system noise and interrupts-- turn to Software Volume article 3.3.18 - "Workaround: Using MONTST with the AM-350."

3.2.25
New Software Patches Available

The following list gives a description of the new software patches now available from AMSD. The products affected by these patches are: AMOS/L, AMOS/32 and the AM-324 VME LPR support software.

Patches in the following list include SPNs 256 through 279, and are current through 15 June 1987. As indicated on the list, some patches have already been released, some patches are still in test. See the June Journal Vol. 9, #6 - Software Article 3.2.24 for more information.

The SPN description in the purpose column ends with the software version(s) this patch is intended for.

SPN #	Module	Purpose
256	—	Still in test.
257	—	Still in test.
258 - 268	—	Released in a previous month.
269	—	Used internally.
270	—	In test.
271 - 272	—	Released in a previous month.
273	DIRSEQ	Ensures that BADBLK.SYS remains as first entry in the UFD, regardless of switches used. AMOS/L version 1.3 and AMOS/32 version 1.0.
274 - 278	—	In test.
279	VLP.DVR	Provides the source code to allow the ability to change to upper and lower case output. AM-324 VME LPR support software.

3.3.15
Warning: DIRSEQ and BADBLK.SYS - SPN-273-00: Patch to DIRSEQ.LIT

DIRSEQ is a program that alphabetizes the file entries in a directory according to the names of the file or by their extensions. DIRSEQ is very useful if you have a large number of files in a directory, because it helps you organize the files. However, it can cause a problem if used on the System Operator's Account, OPR: (DSK0:[1,2]), if there are a large number of files in that account.

The BADBLK.SYS file must reside in OPR: and must appear within the first block of the User File Directory (UFD). Since an account with more than 42 files will take up more than one disk block for the UFD, it is possible for BADBLK.SYS to be sorted out of the first UFD block. Depending on the number of bad blocks and which ones they are, the UFD block containing the entry for BADBLK.SYS may fall beyond one of those bad blocks. If that happens, the next time the system is rebooted or the disk is mounted, the file structure on the disk will be damaged.

During the boot process and prior to loading BADBLK.SYS, the file structure must be intact. This is why disk drive track zero on Alpha Micro computers must certify flawlessly. When the system loads BADBLK.SYS, all of the blocks on the drive, from the first bad block on, are remapped so that those blocks taken out of play do not create holes in the contiguous space on the disk. If the link of the first UFD points to a block which has been remapped, the boot process, unaware of this remapping, will go to the wrong block and fail to find BADBLK.SYS. At this point, not only will BADBLK.SYS and anything else in the second and following UFD blocks be inaccessible, all of the other directories on the entire physical drive will be inaccessible.

The extent of problems can range from no trouble to a total loss of data. If BADBLK.SYS contains zero bad blocks, no problem will occur. If the first bad block does not appear before the second logical surface, there still will not be a problem. But if the first bad block does appear before the second logical surface, then the file structure on at least part of that surface and all of the rest of that drive will be corrupted.

This has not been considered a serious problem until recently because of the multitude of special circumstances that have to occur simultaneously to create the problem:

1. There must be more than 42 files in [1,2].

2. When DIRSEQ is executed, the BADBLK.SYS entry must end up in a UFD block outside of track zero.

3. The UFD must have to be placed beyond the first bad block on the disk.

4. The system has to be booted or the device remounted.

However, now that we have higher density disk drives and because we recommend that the .DIR files used with the VERIFY command remain in the OPR: account, these special circumstances are more likely. For this reason, we have developed the following patch to DIRSEQ.LIT which will cause BADBLK.SYS to remain as the first entry of the first block of the UFD.

As with any patch, it is important for you to enter it exactly as presented or the patch will fail to complete. Specifically, the TYPE statement at label 15$ on the

3.3.15 (Continued) Warning: DIRSEQ and BADBLK.SYS - SPN-273-00: Patch to DIRSEQ.LIT

second page must appear as shown. That is, uppper and lower case are significant, as are the number of spaces.

SPN-273-00
Patch to SYS:DIRSEQ.LIT
All AMOS/L Versions of 1.3
All AMOS/32 Versions of 1.0

This patch causes DIRSEQ to insure that BADBLK.SYS is always the first entry of account [1,2] regardless of DIRSEQ switches used.

Installation instructions:

1. LOG to SYS: and from command level type:

   DIR/H/V DIRSEQ.LIT **[RET]**

   Compare the existing hash total and version with the beginning hash and version listed in step 5 to insure this patch has not already been installed. If it has not, proceed with the patch.

2. If one does not already exist, create a disk account to be used for patch files.

3. Create an AlphaVUE file and name it DIRS01.M68. Enter the patch text shown at the end of this article.

4. Create a copy of DIRSEQ.LIT in the account where you made DIRS01 (i.e., COPY = SYS:DIRSEQ.LIT).

5. Enter the command:

   PATCH DIRSEQ WITH DIRS01 **[RET]**

   You will see the patch file being assembled and installed. If any error messages appear, check to make sure you have entered the patch file correctly and are trying to patch the correct version of the program. If the patch file appears to have been entered correctly, and error messages still appear, please contact the Technical Support Group, (714) 641-7608 for further information.

   Old hash and version:
   020-210-051-570 1.0(106)

   New hash and version:
   077-752-560-221 1.0(106)-1

6. After the patch has been installed correctly, copy the patched version of DIRSEQ.LIT into SYS:.

3.3.15 (Continued)
Warning: DIRSEQ and BADBLK.SYS - SPN-273-00: Patch to DIRSEQ.LIT

```
;Patch #1 to SYS:DIRSEQ.LIT Version 1.Ø(1Ø6)
;ALL AMOS/L 1.3 and AMOS/32 1.Ø
;SPN-273L
;Copyright (C) 1987 - Alpha Microsystems

        COPY    PATCH
BASE:
        OHASH   Ø2Ø,21Ø,Ø51,57Ø
        NHASH   Ø77,752,56Ø,221
        OVER    1,Ø,Ø,1Ø6.,Ø
        NVER    1,Ø,Ø,1Ø6.,1

        .=754
        JMP     PCH1Ø
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
P1Ø:
        .=117Ø
        JMP     PCH2Ø
        NOP

CLRBMP:
        .=12Ø6

CLRBLK:
        .=1352
        JMP     PCH3Ø
        NOP
        NOP
        NOP
        NOP
        NOP

P3Ø:
        .=2612

PCH1Ø:  BCLR    #6,D5
        CMPW    -2(A4),#ED.OPR
        BNE     15$
        BSET    #6,D5
15$:    TYPE    <Sequencing [>
        JMP     P1Ø
```

3.3.15 (Continued)
Warning: DIRSEQ and BADBLK.SYS - SPN-273-00: Patch to DIRSEQ.LIT

```
PCH20:    CMPW    @A3,#3100
          JLO     CLRBLK
          BTST    #6,D5
          JEQ     CLRBMP
          CMPW    @A3,#[BAD]
          JNE     CLRBMP
          CMPW    2(A3),#[BLK]
          JNE     CLRBMP
          CMPW    4(A3),#[SYS]
          JNE     CLRBMP
          MOVW    #3077,@A3
          CLRW    2(A3)
          CLRW    4(A3)
          JMP     CLRBMP

PCH30:    BTST    #6,D5
          JEQ     5$
          CMPW    @A2,#3077
          JNE     5$
          MOVW    #[BAD],@A2
          MOVW    #[BLK],2(A2)
          MOVW    #[SYS],4(A2)
5$:       MOV     #3,D7
10$:      MOV     (A2)+,-(SP)
          SOB     D7,10$
          SUB     #14,A2
          BTST    #6,D5
          JEQ     P30
          CMPW    @A3,#3077
          JNE     P30
          MOVW    #[BAD],@A3
          MOVW    #[BLK],2(A3)
          MOVW    #[SYS],4(A3)
          JMP     P30

          END
```

3.3.16
ISAM and FLOCK

by: Jeff Kreider, Support Specialist
Technical Support Group
Alpha Micro Service Division

[Note from the Editor: This article contains information on using FLOCK to provide ISAM file locking in AlphaBASIC programs in special situations when LOKSER cannot be used. However, please note that as of AMOS 2.0, file and record locking will be an integral part of the operating system, and FLOCK and XLOCK will no longer be supported. For compatibility with AMOS 2.0, programs should be converted now to use the current AlphaBASIC LOKSER syntax.]

Last year we published a series of articles about the current ISAM and how to use it with LOKSER. (See Software Volume Articles 3.3.9 through 3.3.13.) It was quite detailed and covered aspects of ISAM, including details on its internal organization, to a degree not found in other documentation published by Alpha Micro. It did not go into details on getting around the programming protocol required nor did it discuss methods of using other file locking mechanisms. The purpose of the articles was to underscore the need to make existing applications work with LOKSER and to provide insight on the magnitude of work those modifications would require if the application was to be made compatible with AMOS 2.0 and its file locking requirements.

One problem with making a conversion is, if LOKSER is active on the system, it will report errors in the access sequences whether or not the ISAM files are protected. If a software system contains applications provided by different vendors or are maintained by different people within the organization, making the conversion modifications has to be carefully coordinated because LOKSER cannot be used until all software applications are up to standard.

Consider the situation in which the application has been upgraded to support LOKSER, but LOKSER cannot be installed on the system because of another ISAM application that has not yet been converted. So, the programmer uses FLOCK according to the Alpha Micro documentation (See Chapter 6 of the <u>ISAM System User's Guide</u>). However, when executing the program, the program bombs out with FLOCK error 101 "File-channel number is not open in AlphaBASIC for RANDOM processing (Actions 3-6)."

The documentation in Section 6.3.2.2 of the <u>ISAM System User's Guide</u> is incorrect for FLOCK. That is the bad news. The good news is that using FLOCK with an ISAM application is a lot simpler than the documentation implies. The problem is that an ISAM file is **two** files opened in AlphaBASIC under **one** channel number and FLOCK will get confused if used to control OPEN statements. (XLOCK will not become confused because it does not cross check the file channel numbers.)

The secret when using FLOCK, then, is to recognize that access to the .IDA (data file) is only done by accessing the index (.IDX) file via the ISAM statement sequences. Controlling the index controls the data access. This boils down to using two FLOCK calls:

```
LOCK:
   XCALL FLOCK,0,2,{return-code},{channel}
   RETURN

UNLOCK:
   XCALL FLOCK,1,0,{return-code},{channel}
   RETURN
```

3.3.16 (Continued)
ISAM and FLOCK

Immediately prior to any ISAM call that locks the index (ISAM codes 1, 2, 7 or the LOCK #channel), make a call to LOCK. Immediately following any call that releases the index (READ, WRITE, ISAM 3, ISAM 6 or UNLOKR #channel) make a call to UNLOCK.

The LOCK call asks FLOCK for an Action 0 with Mode 2, a request to open a file for exclusive use and wait until the request is granted. The UNLOCK call asks FLOCK for an Action 1, Mode 0, which simply tells FLOCK that exclusive access is no longer needed. This uses exactly the same technique that LOKSER uses at the application level.

Of course, this technique has the same limitations that FLOCK has always had. That is, if a call is not made, the protection will not be there and there is no protection outside of the application that uses it. It also has the same limitations of LOKSER. If not carefully coded, it could lock other users from access to the application longer than necessary. Look at the ISAM/LOKSER Tutorial Parts III and IV (Software Volume articles 3.3.11 and 3.3.12) for details on that aspect.

Below are the two calls to FLOCK and when, within an ISAM sequence, they should be called. These samples are very simplistic in presentation. The tutorial goes into all the detail that you will need on the subject.

```
LOCK:
  XCALL FLOCK,0,2,RET'CODE
  RETURN

UNLOCK:
  XCALL FLOCK,1,0,RET'CODE
  RETURN
```

Adding a Record:

```
CALL  LOCK        ! Request exclusive access and wait
ISAM 1            ! Lookup the key
ISAM 5            ! Get record number
WRITEL            ! Add data record to data file
ISAM 3            ! Add key to index
CALL UNLOCK       ! Release lock on index
```

Deleting a Record:

```
CALL LOCK         ! Request exclusive access and wait
ISAM 1            ! Lookup the key
READL             ! Read and "lock" data record
ISAM 4            ! Delete key from index
ISAM 6            ! Return record number to free list
CALL UNLOCK       ! Release lock on index
```

3.3.16 (Continued)
ISAM and FLOCK

Updating a Record:

```
CALL LOCK        ! Request exclusive access and wait
ISAM 1           ! Lookup the key
READL            ! Read and "lock" data record
WRITE            ! Update record to the disk
CALL UNLOCK      ! Release lock on index
```

The LOKSER syntax should be used (READL, WRITEL, etc.) as documented even though LOKSER is not active. This makes modification easier when LOKSER is activated and prevents unpredictable results from occurring if this syntax is not used.

3.3.17
Introduction to MONTST Problems

by: Jeff Kreider, Support Specialist
Technical Support Group
Alpha Micro Service Division

The MONTST command provides a convenient way to try out a new monitor file, system initialization file or both, by allowing you to boot the computer using those test files. This article discusses why MONTST can fail, and how to work around MONTST problems.

This article contains general information about MONTST problems. Please also see the companion Software article 3.3.18 - "Workaround: Using MONTST with the AM-350," in this issue which gives specific information on working around a MONTST problem that may occur with the AM-350 Intelligent I/O Controller because of that board's multiprocessing capabilities.

An Introduction to MONTST

MONTST assumes there is no activity on the computer system which might interrupt the booting process (that is, that there will be no hardware or software interrupts). In the early days of Alpha Micro, this was a reasonable assumption. It was considered reasonable to require all those connected to the system to stop using the keyboard until the MONTST reboot was complete. As the system became more sophisticated, the probability increased that interrupts would be generated, causing MONTST to fail.

Characteristics of a MONTST Failure

What is meant by the statement "MONTST didn't work"? All it really tells us is, who ever said it did not like what happened when they used MONTST, but it does not tell us what **did** happen. One or two things could happen:

1. You type the MONTST command line on the keyboard using the proper syntax and press the RETURN key. The cursor goes to column one of the next line and sits there; the system "dies." There is no other evidence of activity on the system. The display status is blank. Other terminals are not blinking the "system booting" message.

2. You type the MONTST command line on the keyboard using the proper syntax and press the RETURN key. The cursor goes to column one of the next line. After a moment disk activity is heard. There may be evidence that the test is proceeding, such as the "System booting" message on other terminals. Maybe some of the initialization file appears on the booting job's terminal. Then the system dies.

The first situation we will refer to as a "MONTST failure." The second situation we will call an "incomplete MONTST." Both will be called "MONTST problems."

Causes of MONTST Problems

MONTST will fail if:

1. The monitor being tested is faulty or contains the wrong device driver.

2. The initialization file being tested is faulty.

3. The computer system generates an interrupt during the time the system is reconfiguring itself due to:

3.3.17 (Continued)
Introduction to MONTST Problems

a. Character input.

b. Noise from:
   i. Terminal cables attached to terminals powered off.
   ii. Modems idling.
   iii. Printer cables with Pin 2 connected.

c. Laser printers connected and on-line that use X-on/X-off handshaking.

4. A mismatch between the versions of the software driver for an "intelligent" interface (AM-1013, AM-350, AM-515, etc.) and its microcode.

MONTST will not complete if:

1. The monitor being tested is faulty.
2. The initialization file being tested is faulty.

and may not complete if:

1. Multiprocessing controllers exist in the current configuration and are redefined in the new configuration.

Sometimes an incomplete MONTST reboot will appear as a MONTST failure. However, a failure will only look like a failure, not an incomplete MONTST.

The reason for using MONTST is to verify the integrity of either a monitor or an initialization file. It would be nice to assume that if MONTST failed, it was because either the monitor or the initialization file (or both) being tested had a problem. That is not necessarily the case. The environment is just as likely to be the cause of MONTST problems; this fact is often overlooked.

Evolution of MONTST Problems

The first board we introduced capable of of doing its own local processing was the AM-310, which contained an onboard Z-80 chip. At that time, such a configuration was rare, and seldom caused a MONTST problem to occur.

We soon went to a new CPU (M68000); because of its greater addressing capability, more memory could be added to the computer, allowing more users to run on the system. This made it more difficult to coordinate keeping everyone off the system until the system rebooted. To get around that, Alpha Micro took advantage of the LOCK Keyboard feature of the newer terminals to limit the possibility of extraneous keystrokes interrupting the CPU during critical periods of the MONTST reboot.

This feature made MONTST problems less likely. However, it only shortened the list of causes of MONTST problems. Soon we added more terminal interface boards that had their own processor (e.g., the AM-1003 I/O expansion board which has a Z-80 controlling two of its ports). We also introduced the S-100 counterpart for AlphaRJE support, the AM-330 Communications I/O board. Most recently, we have the Herbie technology hardware that also contribute to MONTST problems.

Another side effect of the faster CPUs (even more so with the M68020) is that they are more sensitive to noise, which can cause the system to malfunction. Although it can cause a MONTST reboot to fail, noise is an indication of another problem, not a MONTST error. The fact remains noise is another factor to consider when tracking down MONTST problems.

3.3.17 (Continued)
Introduction to MONTST Problems

Bad Monitor or .INI File

If the monitor is bad or is using the wrong device driver for the system on which it is tested, MONTST will fail. The monitor can be verified with the MONHSH program and compared with the current patch level.

If a faulty initialization file can cause MONTST to fail or, at least, appear to fail. If, for example, the user partition is too small, or the initialization file is too large, the MONTST reboot will complete but it will leave a 12 on the status display. This situation occurs most commonly when a change has been made to the initialization file to add a user or peripheral device. These changes make the initialization file too big to be loaded into the bootup job's memory partition and the monitor thinks it cannot find it. Sometimes, adding more comments to the initialization file is all it takes to cause this problem.

Sometimes, the :R or :T at the beginning of the initialization file inadvertently gets trashed. This causes the initialization file display to default to :S mode and suppresses all terminal output. This can make it appear that the system failed to boot.

These kinds of problems are typical of MONTST failure and are where the investigation in such cases centers. Such causes can be illusive enough in themselves. Complicating the picture are the new factors of multiprocessing and faster CPUs, causing more MONTST problems today than in the past. Recognizing that fact is an important step in isolating the cause of a MONTST problem.

MONTST and Interrupts

Terminal service and disk service software rely on interrupts to indicate the hardware has data available or is ready to take more data. Rather than having the monitor constantly polling these devices for their needs, which would significantly reduce system throughput, the hardware generates interrupts. An interrupt causes the CPU to immediately jump to a predesignated area in memory to start execution of a section of code called the "interrupt handler."

When an interrupt from a terminal I/O port is generated, the hardware will cause execution to jump to the interrupt handler. The interrupt handling routines are set up during the execution of the TRMDEF statements in the system initialization command file. The MONTST program loads the monitor being tested from the disk into memory and then overlays the existing monitor with the "fresh" one. During this process, the link to the interrupt handling routines is lost.

If any port generates an interrupt after this time and before the TRMDEF statement which redefines this port is executed, no interrupt handler can be found to take care of the interrupt. The result is unpredictable, but it is usually fatal to the system bootup.

Disk hardware sends interrupts upon completion of a task. However, disk service hardware doesn't cause problems with MONTST the way the terminal service system does because disk interrupts are a result of disk requests made by AMOS, so they are controllable and expected. Terminal input interrupts are initiated by the outside world; therefore, they are uncontrollable.

MONTST and Noise

When a character is received by the interface port, an interrupt is generated and the above problem may occur during a

3.3.17 (Continued) Introduction to MONTST Problems

MONTST reboot. Actually, anything received by that port will generate the interrupt, even if it is not a valid character. Noise on the cable will also be perceived by the interface card as input and will generate an interrupt. In addition, noise can propagate throughout the entire system, causing malfunctioning of any system component.

If a long cable is connected to a terminal, and the terminal is powered off, this leaves the cable unterminated, causing the cable to act as an antenna. This allows the cable to pick up noise from external sources such as fluorescent lights or other electromagnetic radiation. This noise can appear to the terminal interface I/O card as character input and generate an interrupt.

Modem cabling can generate the same kind of noise even if cabled correctly. Modems causing MONTST to fail have a variety of configurations and only some can be changed to still allow the modem to function. Some modems will cause a MONTST problem if powered on but not when off. Others will cause a problem only when powered off. Still others will cause a problem in either state. Technical Support has had evidence that MONTST may not fail all of the time when modems are hooked up, regardless of configuration.

The most common cabling problem is caused by hooking up Pin 8 to some of our I/O cards. The AM-300, for example, does not connect to pin 8, the consequence is the introduction of noise on the interface port which will "leak" into the board and be serviced as an interrupt. Another example is having Pin 2 wired from a printer that does DTR handshaking. Pin 2 is an input from the printer to the I/O port and is not necessary for communications or protocol when using DTR; in fact, it is a source of noise.

Any printers with X-on/X-off handshaking (especially laser printers) create an interesting complication in this situation. X-on/X-off handshaking is CPU intensive to begin with and will generate interrupts upon reception. If you have the option, use DTR handshaking; otherwise, taking the printer off line may avoid the MONTST problem. However, it may introduce the noise problem.

To isolate the cause of the problem to extraneous interrupts for terminal service, the easiest thing to do is to disconnect all of the terminal cables from the back of the chassis except for the main boot job. If MONTST will now work successfully, then the problem is due to one of those causes. At this point, each cable can be connected back, one at a time, until the problem materializes again. The solution to this problem (and even its isolation to some extent), really depends on the freedom you have to rearrange things.

MONTST and Special Hardware

Customers have called the Technical Support Group to report problems with MONTST for boards that contain their own CPU such as the AM-310, AM-1013, AM-1213, and AM-330.

More recently, the new AM-350 Intelligent I/O Controller has also had problems with MONTST because of its special multiprocessing capabilities. Other intelligent controllers such as the AM-515 and AM-520 have not had the same sort of problem because they are part of the disk service system, which is much less vulnerable to MONTST problems. (See the "MONTST and Interrupts" section above for reasons why terminal service is more likely to cause MONTST problems than disk service hardware.)

3.3.17 (Continued)
Introduction to MONTST Problems

The multiprocessing, intelligent controllers will cause two kinds of MONTST problems:

Initialization Problem:

Currently only a hard reset will shut these boards off. They are reinitialized by the microcode for the device during the time the system is being reconfigured.

The problem exists because MONTST does not reset the "intelligent" hardware connected to the system. Because the hardware is still "live," it is capable of sending interrupts even thougn the operating system no longer knows about the hardware. With AMOS 2.0, MONTST will reset the SIO chips to an inactive state, which will prevent them from creating interrupts until after the chip is reinitialized. But, this will not eliminate interrupts due to noise from power supply wiring or dangling terminal cables leaking into the ports.

The best workaround for this situation is to avoid it. Configure the system to hard boot from a minimal intialization file that does not address these kinds of devices. This way, when the reset button is pressed, whatever hardware is on the system will be reset and require software reinitialization to again become functional. If the system initialization does not address these boards, then they cannot generate interrupts that will interfere with MONTST.

Once the system is running from this minimal configuration, using MONTST to test another monitor and/or another initialization file should not cause the problems discussed above.

Mismatched Driver and Microcode:

The other problem associated with "intelligent hardware" is a mismatch between the device driver and its microcode. This will not cause MONTST to fail, but it will cause the boot to be incomplete. An example was published in a recent article when 1.3(123) was released. We shipped an incompatible version of the AM1013.MIC file for the AM1013.IDV. This caused systems to die on the TRMDEF line that initialized that port [See Software article 3.1.10 - "AM-1013 on AMOS/L 1.3B" in the September/October issue of the AMSD Journal.]

More recently, patches to the AM-515 support software complicated this situation by using the patched version of the microcode with a different version of the driver.

Consider the situation where the system device is controlled by the AM-515 and new versions of the 515DVR.DVR and AM515.MIC are down loaded to the system. According to standard practice, when generating a new monitor, MONTST is typically used to make sure the monitor works prior to renaming it to AMOSL.MON (or AMOS32.MON).

Because MONTST will not reset the AM-515, it will remain initialized with the previous version of the microcode. When MONTST loads the monitor and tries to execute it using the new driver, it will fail to communicate with the hardware because of the mismatch between the new driver and the old microcode. MONTST will appear to fail. In reality MONTST worked just fine, but the system initialization failed to complete.

At this point AMOS??.MON, which still has the old driver, will fail to boot on a hard reset. When the reset button is pressed, the AM-515 is reset and requires reinitialization. After the monitor is loaded and it recognizes that the AM-515 needs to be initialized, the new microcode is loaded from the disk and it initializes the board.

3.3.17 (Continued) Introduction to MONTST Problems

Then the boot process tries to communicate with the old driver in the monitor with the AM-515 initialized with the new microcode; the system will fail to boot.

At this point, the system is dead and can only be recovered by a warm boot that contains a matched driver and microcode set. There is no other way around this. The new driver must be incorporated into the AMOS??.MON by using the MONGEN command, and the reset button has to be pressed to bring up the system. For this reason it is important to have a reliable warm boot tape available because MONTST cannot be used in this situation.

Summary

There are many reasons why a MONTST will fail to perform as desired. In troubleshooting such situations, it is important to recognize many of the reasons for the apparent failure are unrelated to the integrity of the files being tested.

This article has discussed general information about MONTST and why it can fail. More detailed information on working around a specific problem with using MONTST on a computer containing the AM-350 Intelligent I/O Controller is provided in the Software article 3.3.18 - "Workaround: Using MONTST with the AM-350," in this issue.

3.3.18 Workaround: Using MONTST with the AM-350

by Ray Ezell, Systems Programmer
Advanced Products Division

This article discusses a potential problem with using MONTST on a computer with an AM-350 Intelligent I/O Controller board installed, and presents a workaround for the problem. This information is applicable to both Phase I and Phase II of the AM-350 software.

A companion article in this issue, Software article 3.3.17 - "Introduction to MONTST Problems," contains general information on why MONTST can fail, and gives guidelines for troubleshooting such problems. That article also gives more detailed background information on why noise and interrupts cause MONTST problems.

The Problem

The problem with using MONTST on a computer with an AM-350 installed exists because MONTST reconfigures (reboots) the system, but does not reset the hardware connected to the system.

You have seen this problem if you ever used MONTST to test a system initialization command file that didn't redefine all of the I/O ports that were defined in the original initialization file and then typed a key on a terminal connected to one of the ports that was no longer defined. In most cases, the computer "locked up" and the bootup failed. The reason this problem occurred was because the I/O hardware was still initialized and capable of generating interrupts to the main CPU even though the hardware was undefined as far as the operating system was concerned.

In the past, it was rare for undefined hardware to try to "talk to" the operating system, so this did not cause a problem very often. However, with the advent of the AM-350 and other Intelligent Controllers, the problem is more complicated because the AM-350 software is directly involved with the internal operation of AMOS.

Future Plans for MONTST

In the AMOS/L and AMOS/32 2.0 software releases, MONTST will be changed to allow hardware reset, which will take care of this problem on AM-350s.

Although only the AM-350 driver will be changed to take advantage of MONTST's new capability, any type of hardware driver can be changed to do so by linking its own self-defined reset routine into a chained calling sequence during driver initialization, which will allow the driver to reset the hardware during the MONTST. (Details on this procedure will be provided to systems programmers with AMOS 2.0 documentation in the future.)

AM-350 Complications

In the general case, you can eliminate a problem with MONTST by disconnecting the terminal cables at the computer end connected to ports not defined in the test initialization file.

However, when an AM-350 is present, it becomes more complicated because the AM-350 software continually monitors certain states of the operating system even if there is no I/O activity occurring through its ports.

When an AM-350 has been initialized via execution of a TRMDEF from the system initialization command file, it is "active" even in its idle state when it is not processing any I/O requests. During this act-

3.3.18 (Continued)
Workaround: Using MONTST with the AM-350

ive idle state, the AM-350 examines the Terminal Control Blocks (TCBs) defined in system memory for each of its initialized ports in an effort to ensure it has honored all requests for I/O.

The problem stems from the fact that there is no way the operating system can inform the AM-350 that a MONTST is taking place. Consequently, depending on the new system configuration and the data that is overlayed on top of the area in system memory where the AM-350 TCBs were previously located, the AM-350 may misinterpret the new data in the previous TCB locations as an I/O request and attempt to honor the pseudo request. The affect on the computer is unpredictable, ranging from garbage on the terminal to system lock up.

Problem Workaround

The only situation consistently causing a MONTST failure is when the monitor and initialization file being tested are the same as those currently in effect except a TRMDEF statement that previously defined the first AM-350 port now defines a non-AM-350 port.

The best workaround for this situation is to avoid it. Configure the system to hard boot from a minimal initialization file that does not address the AM-350. This way, when the reset button is pressed, whatever hardware is on the system will be reset and require software reinitialization to again become functional.

Once the system is running from this minimal configuration, using MONTST to test another monitor and/or another initialization file should not cause problems.

If using a minimal initialization command file is not practical, and for some reason you do want to change the interface definition in the first AM-350 TRMDEF statement in your standard system initialization command file to a non-AM-350 interface, you can usually prevent a MONTST problem by adding or deleting a job definition from the JOBS and JOBALC statements in the system initialization command file. This causes the TCBs defined in the new configuration to be slightly offset from the positions of the TCBs in the original configuration, which means the AM-350 is less likely to become confused by spurious I/O requests. You need only be concerned with the first AM-350 TRMDEF statement, because it will cause the AM-350 to be reset.

In general, then, a MONTST will usually be successful when using:

1) A different monitor version because the monitor size is probably different, which causes the TCBs to be offset, or;

2) A system initialization command file that is different before the first AM-350 TRMDEF statement than the initialization file in effect.

Even so, the only way to absolutely ensure a successful MONTST is to preserve the position of the first AM-350 TCB and maintain it is an AM-350 TCB.

Using MONTST to Test a Warm Boot File

Similar problems can occur when using MONTST to test a warm boot file if the AM-350 is defined at the time of the MONTST, but the warm boot file specifies a non-AM-350 interface for the system I/O port. This depends on the data patterns within the warm boot monitor that overlay the area in memory previously allocated to AM-350 TCBs.

3.3.18 (Continued)
Workaround: Using MONTST with the AM-350

Unfortunately, the workaround for using MONTST to test a warm boot file is not as straightforward as the procedure for using MONTST to test a system initialization command file. You could try recreating the warm boot file by loading the various modules in a different order, but this could be laborious. Probably the simplest solution is to "kill" the AM-350 board.

The only method that currently exists to "kill" the AM-350 once it has been initialized is to perform a hardware reset. Unfortunately, if the AMOSL.INI or AMOS32.INI file defines the AM-350, the AM-350 will be re-activated. Therefore, if you want to test a system configuration in which the AM-350 is not included, but your system initialization command file defines the AM-350, you must follow the steps below. We use AMOS??.INI to mean either the AMOSL.INI or the AMOS32.INI, depending on which CPU is in use.

1. Copy the AMOS??.INI into TEST.INI.
2. Remove TRMDEFs defining the AM-350 from the TEST.INI.
3. Increase the number of jobs defined in the JOBS statement by one.
4. Make other changes as required by the removal of the TRMDEFs.
5. Use MONTST to test TEST.INI.
6. If the system fails to come up or come up completely, investigate and resolve the problem from the information provided in the Software article 3.3.17 - "Introduction to MONTST Problems" in this issue, and go to step 5.
7. If the MONTST was successful save AMOS??.INI into AMOS??.SAV.
8. Copy TEST.INI into AMOS??.INI.
9. Press the reset button.
10. Use MONTST to test the new configuration or warm boot file.
11. Upon completion of the test, copy AMOS??.SAV to AMOS??.INI.
12. Press the reset button or use MONTST to restore the normal system configuration.

Summary

When using MONTST on computer systems with AM-350 Intelligent I/O Controllers installed, the location of the first AM-350 TCB in the old and new configurations must be preserved or the reboot will fail. If the AM-350s are to be eliminated from the new configuration, offsetting the TCB for another interface from the location of the old TCBs of the AM-350 will generally be successful. If MONTST is not entirely successful, following the above AMOS??.INI modification should provide a sure method to provide an appropriate environment in which MONTST can function.

Remember that this problem will be solved for AM-350 installations with the AMOS 2.0 release.

6.4.4 Warning: Information on Image Mode Backup for VIDEOTRAX Users

VIDEOTRAX has two modes of backup and restore: file mode and image mode. Image mode is faster, especially under older DOS versions 2.x or when using a VCR other than the VIDEOTRAX remote control VCR. However, image mode is also more susceptible to problems caused by hardware that is not 100% IBM compatible and software that is not 100% MS-DOS or PC-DOS compatible. This article contains information on when it is **not** appropriate to use image mode.

Large Disks

Several software packages are on the market that allow DOS disk volumes to be larger than 32Mb. Typically, this is done by making the sector size on the disk larger than 512 bytes. By doubling it to 1024 bytes, DOS can support up to 64Mb in one disk volume. However, when one of these packages is installed, **only file mode backup and restore will work correctly.** Image mode may appear to work, but it will not save the data correctly and you will not be able to restore the data. This limitation is because DOS itself does not support disk volumes larger than 32Mb.

Partially Corrupted Disks

Now and then a situation occurs in which the data on a disk becomes partially corrupted. The disk may not be bootable, although it can be accessed when the computer is booted from a floppy diskette. Certain files may be inaccessible while others seem to be just fine. It may be the disk contains data that has been updated since the last full backup, and that this data is valuable. Under these circumstances, when you want to retrieve some of the data from the disk, **only file mode backup and restore will work correctly.**

Image mode may appear to work, but it may not save the data correctly and it may not be able to restore it correctly after the disk has been reformatted.

Summary

In order to avoid problems when using image mode, the disk must be a standard DOS volume of 32Mb or less and it must be correctly formatted and initialized. If you have any suspicion that there may be problems with a particular disk, perform a file mode (not image mode) backup before you reformat the disk.

> **It is a good idea to run CHKDSK before any backup, but it is especially important to do so before an image backup.**

For more information or support on the operation of VIDEOTRAX, please contact Alpha Micro's Technical Support Group at (714) 641-7608.

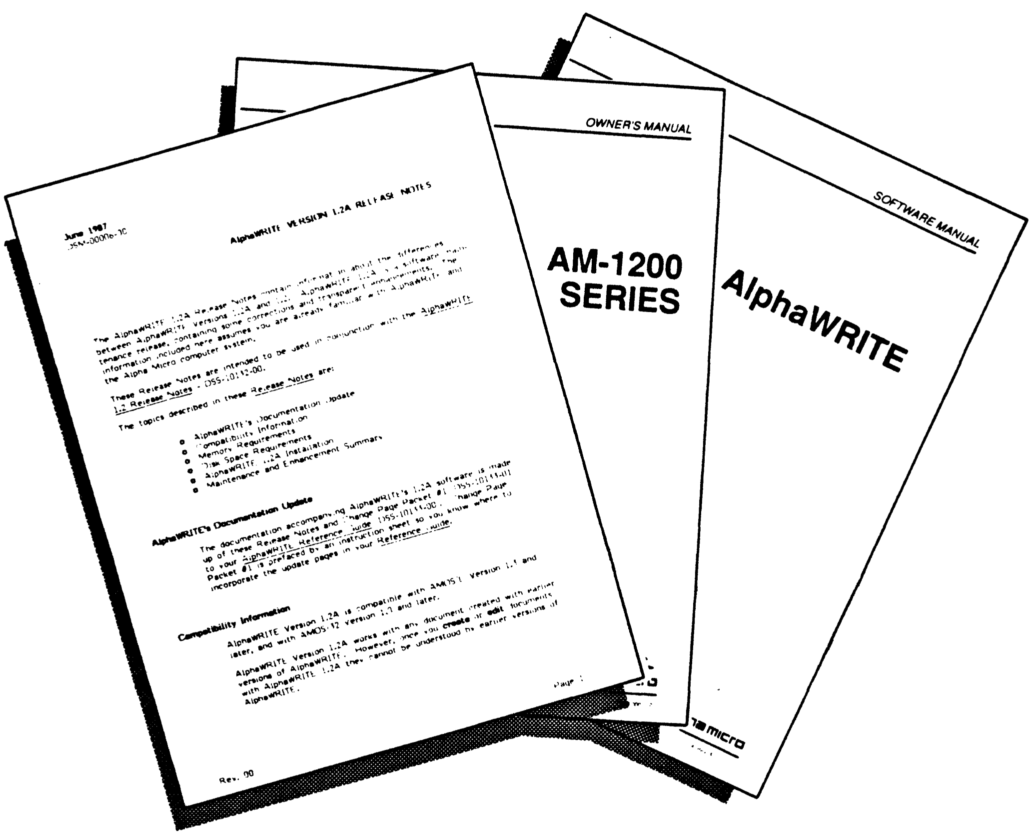
3.2.6
New Documentation Releases

Several new user manuals are available for sale in July. (See the July Alpha Micro Reseller Price List for prices.) The new documents are:

AlphaWRITE 1.2A Release Notes
DSO-00006-00
The Release Notes describe the changes between AlphaWRITE version 1.2 and 1.2A.

Change Page Packet #1 to the AlphaWRITE Reference Guide
DSS-10133-01
This change page packet updates the existing AlphaWRITE documentation to reflect changes introduced by the latest 1.2A release of the software.

Change Page Packet to Version 00 of the AM-1200 Owner's Manual
DSU-00011-01
Documents minor changes to the self test display and the characteristics and use of an SCSI (Small Computer System Interface) Winchester disk drive.



AMSD JOURNAL TABLE OF CONTENTS UPDATE PAGES

The next pages of the Journal are updated Table of Contents pages for your back issue volumes. The updated pages are:

SOFTWARE INFO. VOLUME:	- Section 3 - AMOS/L Operating System (2 pages)
	- Section 6 - Software Packages
	- Section 8 - AMOS/32 Operating System (2 pages)
GENERAL INFO. VOLUME:	-Section 3 - Manuals

- A footer line at the bottom of each table of contents page shows you revision information. This line shows month, year, volume and issue number of the Journal this table of contents page arrived with.

- All table of contents pages have a title line showing which volume they belong in: Hardware, Software or General Information.

- Entries for articles published since 1983 show the month and year of publication.

- Cross reference article entries use this format:

  "Article Name" - Cross reference: See Volume Name Article #.#.# - [Month Year]

  Where Volume Name is Hardware, Software or General Information. Where #.#.# is the article number designating section, category and article number. (For example, article 6.4.3 is filed in section 6, under category 4 and is the 3rd article in category 4.) [Month Year] is the Journal publication date for the article.

Section 3 - AMOS/L Operating System (continued)

Section 3 - AMOS/L Operating System (continued)

3.3 AMOS/L Utilities

3.4 Programming Information

3.5 Program Hash Totals

SECTION 6 - SOFTWARE PACKAGES

SECTION 8 - AMOS/32 OPERATING SYSTEM

8.1 AMOS/32 Releases

8.2 AMOS/32 Patches

Section 8 - AMOS/32 Operating System (continued)

8.3 AMOS/32 Utilities

8.4 Programming Cautions

8.5 Program Hash Totals

SECTION 3 - MANUALS