BUGLIST #18 (Version 4.5) 09/01/81

Releases and sub-releases since last buglist :

4.5 --- AlphaMAIL, AM120, 11 New Monitor Calls, TDV Driver Writer
4.5(1) --- Patch to 4.5 given in Software Notes II-4 (06/17/81).
Changed was DSKANA, and the version number in SYSTEM.MON.
4.5A --- Sub-release for Winchester systems (AM420) (next buglist)

AMOS 4.5 came out in the middle of May 1981.

My humorous instincts have placed heavy pressure on me to change the titles of the 3 buglist sections to "bugs,suggs, and ushs"; well, maybe later ...

Some significant problems with 4.5 have surfaced; most have fixes.

Bob Fowler / A. I. S. / 800 San Antonio Ave / Palo Alto, CA 94043

AMOS BUGS (Version 4.5)

(1) BASIC documentation --- reserved words were not updated

Appendix C in the 4.5 BASIC manual should be updated to contain the new reserved words APPEND, FORCED'RANDOM.
[thanks, Gerry]

(1) BASIC documentation --- MEM table errata

The table of MEM function values is incorrect as follows :

entry	says	should be
MEM(7)	dummy data	array indices
MEM(8)	array indices	variable storage
MEM(9)	variable storage	dummy data

(1) BASIC MAPS and continuations --- don't work well past 250 bytes

It is convenient to initialize large (eg, 256-byte) string variables, using the continuation syntax character "&". For example, consider a string initialized with 50 bytes on each line, like this :

```
10 MAP1 A$,S,400,"123456789.123456789.123456789.123456789.123456789."+&
"123456789.123456789.123456789.123456789.123456789."+&
"123456789.123456789.123456789.123456789.123456789."
```

This is a 3-line version; in what follows, we consider versions that contain up to 10 or more lines, by replicating the middle line.
A 10-line version LOADed into BASIC will set "source line overflow".
An 8 or 9-line version will LOAD ok, but hangs up job if RUN.
A 6 or 7-line version will LOAD, but a RUN will report "out of memory".
In a 32K user partition, there are actually over 10K bytes left.
A 5-line version will pass RUN, but a subsequent PRINT A$ will show that A$ is now missing its 1st,2nd,239th, and 240th bytes !!! Try it.
A 4-line version has no problems, but is only 200 bytes --- not fair !
[thanks, Gerry]

(1) RUN --- branching out of FOR loop leaves control variable "sensitive"

Branching out of a FOR loop is known to consume 18 bytes of memory.
However, if another FOR loop is later run using the same variable,
then BASIC evidently begins to mis-interpret the pseudo codes.
This problem did NOT occur in 4.4; it began with the 4.5 RUN.
The simplest example appears to be the following :

```
10   FOR I=1 TO 1 : GO TO SKIP : NEXT
20 SKIP:
30   GOSUB ROUTINE : PRINT "DONE"
40 ROUTINE:
50   FOR I=1 TO 1 : NEXT
60   RETURN
```

When the above is executed, no output occurs exept the message:

```
   RETURN without GOSUB in line 60 of A.RUN
```

If J (not I) is used in line 50, there are no problems.
Another example of this problem also goofs up at the RETURN command;
no error is given, but in effect a STOP is executed instead of RETURN.
instead of the RETURN command.
[thanks, Ron]

(1) RUN --- INPUT #n and READ no longer support space or tab delimiters

Enter the following program :

```
10 DATA 81 82 91 999 091 092
20 READ A,B,C,D,E,F
30 PRINT A;B;C;D;E;F
```

The output is " 81 82 1 99 91 92 ". Leading 9's are truncated.
Another sample is :

```
10 DATA 1 TEST
20 READ A,B$
30 PRINT A;B$
```

The output is "1 EST". After a numeric field, blanks, string field,
the string has the first character truncated.
The same truncations will happen when using INPUT #1 on an ASCII file.
None of these truncations occur when using 4.4 RUN.
For those of you with nicely formatted data files, there is hope :
as long as 1 comma occurs between each field, no truncation occurs;
(additional spaces and/or tabs on either side of each comma are ok).
This can take awhile to diagnose and track down ...
[thanks, Gerry and Jerry]

Addendum : Software Notes for July 1981 (arrived 08/07/81) is humorous.
It says that pre-4.5 INPUT "did not handle inputting consistently",
which is not true --- pre-4.5 allowed anything that was not ambiguous.
Effectively, all string inputs had to end with a comma or a return.
It then says that all data must be separated by commas, except numeric.
This is also not true, due to the bug described above (eg, 81 82 91 92).
Finally, the last (fifth) example given is not correct (try it).

A fix to the truncation of leading 9's, comes from AM over the phone :

```
DDT RUNSML
23260/  BHIS 23266   BHI 23266
DDT RUNLRG
26136/  BHIS 26144   BHI 26144
```

I'll bet that a similiar fix is also possible for the string truncation.
We may have a case here of a bug being turned into a feature.
Even if this change was indeed intentional, rather than accidental,
AM still implemented it incorrectly, and didn't originally document it
even though it was imperative to do so because of pre-existing software.
[thanks, Compuwest]

(1) RUN --- problem with un-MAPped/un-DIMed arrays

Create, COMPIL, and RUN the following program :

```
10 ! DIM B(10)      ! WORKS OK IF THIS INCLUDED
20 ! C=1            ! WORKS OK IF THIS INCLUDED
30   B(C+1)=1
40   FOR D=1 TO 1
50   C=1            ! WORKS OK IF THIS DELETED
60   NEXT
```

It gives the error : NEXT without FOR in line 60 of (filnam).RUN
RUN is evidently having trouble allocating the array at RUN time.
This problem will happen even if line 30 is NEVER EXECUTED !!!
To demonstrate, insert these two lines and COMPIL and RUN.

```
15   GO TO A
35   A:
```

[thanks, Lai]

(1) RUNSML --- careful of accounting goodies

The new small RUN package has no transcendental functions (no powers).
Thus, at least 1 line in the accounting package will give you problems.
In PRXPND.BAS, line 2310, a stall is executed by doing the following :

```
FOR ZZ=1 TO 80 : GG=(2^5) : NEXT ZZ
```

This was the only exponentiation left in our Accounting programs,
but we have probably removed others over the past year+.

(1) COMPIL --- ++INCLUDE's mess up error reporting

Consider the following program with 1 INCLUDE and 1 missing label (A) :

```
++INCLUDE B.BAS
20 GO TO A
30 END
```

Where B.BAS contains one line of code, for example :

```
10 PRINT
```

COMPIL will detect the error, but blames it on line 30.
If line 30 is removed, the error is blamed on a blank line.
If the INCLUDE line is removed, the error is correctly reported.

In general, it appears that if x lines of code are INCLUDEd, and a pass 2 error occurs on the y-th line, the error will be blamed on line x+y.
If the program is shorter than x+y lines, a blank line will display.
[thanks, Gerry & Lai]

(1) VUE --- problems using SBLK with REPLACE and GLOBAL

Create a file containing 4 lines of "A A" in VUE, then do exactly this :

(a) mark off first 2 lines (using control-P)
(b) go to menu and do an SBLK command
(c) do a REPLACE command, changing "A" to "B"
(d) type "Y" responses; only 3 REPLACEs are found !

Bug #1 --- VUE fails to find an SBLK match at the very end of the block

(e) CLEAR the markers
(f) set new markers on the 3rd and 4th lines
(g) Do a GLOBAL command, changing "C" to "D"; VUE makes 3 changes !
    Effectively, it GLOBALs "A" to "D" instead of "C" to "D".
    As in Bug #1 above, VUE fails to find the last (pseudo-) match.

Bug #2 --- After doing a REPLACE with SBLK, and the markers are reset, a subsequent GLOBAL will sometimes use the same match as the REPLACE, ignoring the string actually entered by the user.
[thanks, Gerry]

(1) VUE --- if first line UNYANKed is blank, it doesn't set UNYANKed

Create a 3-line file in VUE containing a blank line + a non-blank line + a blank line. Then UNYANK the first 2 lines to a second file (do control-P on 1st and 2nd lines, position cursor on 3rd line). You will find that the second file is missing the first (blank) line. In general, when the first line UNYANKed is blank, it isn't UNYANKED. When UNYANKing several blank lines, one doesn't get UNYANKed. Remember : "If first line is blank, it don't get UNYANKed".

(1) VUE --- minor bug causes low intensity in command screen

In any file, mark off a section with control-P's, and position the cursor inside the section, then type the following 9 characters (no blanks)
esc A B C D E control-U control-L control-Y
You will now see low intensity.

(1) TXTFMT --- /CENTER (text) too big can crash system

A user reports that if a CENTERed text exceeds the value of LINESIZE-2, that AMOS crashes.

(1) DSPLY.SBR --- problems when numeric output field is on odd byte address

Create the following program and RUN it :

```
10 MAP1 A
20   MAP2 B,S,1
30   MAP2 DECMAL,F,6,500
40 XCALL DSPLY,3,12,1,DECMAL,3,0
```

On my terminal, I get the following output :
415,387,115,880,3 ,i3,,Vs,T.-.
When line 20 is removed, the output comes out correctly, as "5.00".
This took awhile to track down ....

(1) INPUT.SBR --- doesn't dings-dongs on row 12

Enter followings, COMPIL it, LOAD SYS:FLTCNV, LOAD BAS:MESAG, and RUN it.

```
MAP1 ENTRY,S,10
PRINT TAB(-1,0);
XCALL NOECHO
XCALL INPUT,10,1,1,0,"$ ",ENTRY,INXCTL,1
XCALL INPUT,12,1,1,0,"$ ",ENTRY,INXCTL,1
```

If you type 2 characters on row 10 you get beeped at, but not on row 12.

(1) INPUT.SBR --- doesn't check date fields very well

The "D " format command is for entering 6-digit dates in format mmddyy.
The validity checking is confined to the following tests:
all characters must be digits
mm < 13 , dd < 32
The followings are NOT checked and will always slip by :
mm > 00 , dd > 00
mm > 28 for the appropriate months and leap/non-leap years
[thanks, Jan]

(1) MEMDEF --- picky about blanks between parameters

If extra blanks occur within the MEMDEF parameter list, it can fail.

```
MEMDEF 101,3,0      ; is ok, but doesn't visually "line up nice"
MEMDEF 101, 3,0     ; gives error message
```

This should be easy for Alpha Micro to change in MEMDEF.

(1) TRMDEF --- doesn't like trailing blanks after parameters

The following works fine:

```
:T
JOBS JOB1
TRMDEF CRT1,AM300=1:16,SOROC,80,80,40
DEVTBL DSK1,DSK2,DSK3,DSK4,DSK5
BITMAP DSK,1818,0,1,2,3,4,5
SYSTEM
CLKFRQ 60
MOUNT DSK1:
MEMORY 0
```

However, put a blank after "40" in the TRMDEF line, and reset will die.
[thanks, Ray]

(1) DSKPAK --- one bug fixed, another one put in

DSKPAK finally works; that one-bit DDT was incorrorated (Buglist 16).
However, one other bit was changed and should be changed back, thus

```
DDT DSKPAK                  (in 4.5 AMOS)
572/  BNE 612    BNE 602
```

The above instruction is contained in a chunk of DSKPAK code that is moved onto the user stack, and then executed there. It turns off user's bank, turns on bitmap bank, fiddles around, then returns to user's bank. The BNE 602 stays within that chunk of moved code, but not BNE 612; thus, if the branch ever occurs, it executes garbage on the user stack. From what we can decipher, this will crash system whenever the disk being packed shares a bitmap area with another disk, and that other disk was occupying the shared bitmap area when DSKPAK was invoked. The DSKPAK will finish, re-write bitmap properly, crash system, & scare the pants off the system's programmer ("are my files screwed up ?"), But a simple re-boot should be sufficient cure.

(1) SUSPND --- only suspends a job if it is already executing

Consider 2-crt system with CRT1 attached to JOB1, CRT2 attached to JOB2. If JOB1 is executing, then SUSPND/REVIVE JOB1 both work fine from CRT2. If JOB1 is at monitor level, then SUSPND JOB1 effectively does nothing; a SYSTAT will now show JOB1 as suspended ("SP"), but JOB1 will accept an AMOS command (on its crt or FORCEd), and upon exit will REVIVE itself. This can probably be best remedied by changing TRMSER to only pass on a command string if a job is not suspended.
[thanks, Gerry]

(1) MEMORY --- evidently doesn't correctly reset JOBMEM parameters

If 2+ banks are brought up, and SYSTEM.INI's last line is MEMORY 0, then Job #1's bank parameters are evidently not properly set up. If a JOBMEM is subsequently entered on Job #1's terminal, the destination bank will be (correctly) turned on, but the original bank will also (incorrectly) be left on. This leads to buss errors, etc.
[thanks, Gerry]

(1) DDT --- doesn't input CALL or JSR

I just know this must be ancient news to Assembler fans, but here we go. DDT will disassemble CALL just fine, but will input neither CALL nor JSR. I made up a DDT fix and was finally forced to settle for "WORD" commands!

(1) FMT500 --- re-usable, but looks suspicious the second time

A customer of ours does a Hawk backup sequence where FMT500 is loaded into memory, used once ok (all blocks become FMT500'ed to zero bytes), and then used later a second time with the followins funny results : block 0 (the label) from the disk beins FMT500'ed is copied to all 9695 other blocks on that disk. There was also on the label on the first disk beins FMT500'ed, so this is a different kind of result.
The hash code on RES:FMT500 is different from DSK0:FMT500 after usase.
[thanks, Tom]

(1) SYSTEM.MON --- now converts all command input lines to upper case

Create and MACRO the followins prosram (TEST.MAC) :

```
          COPY    SYS
    TEST: TTYL    @R2
          EXIT
          END
```

Type "TEST a" at monitor level. The response is "a" in 4.4, "A" in 4.5
This means that it is IMPOSSIBLE to pass lower case via the input line.
Programs that used this in pre-4.5 must be re-desisned somewhat.
Nothins was mentioned about this in the 4.5 notes.
[thanks, Gerry and Bill]

(1) FIX --- problems with USREND (but not USRBAS nor USRFRE)

Create, VUE, MACRO, SYMBOL, FIX, and sinsle step throush this prosram:

```
          COPY    SYS
          LEA     R0,HERE
          USRBAS  0(R0)     ; always ok
          USRFRE  0(R0)     ; always ok
          USREND  0(R0)     ; disassembles wrons.  prematurely exits.
    ;     USREND  R0        ; this has no problems in FIX
          ADD     R0,R0     ; exits before this instruction
          EXIT
    HERE: WORD    0
          END
```

FIX incorrectly disassembles USREND 0(R0), displayins a "NOP" after it.
Evidently, FIX is confused about the 0 offset, & disassembles it twice.
Also, sinsle steppins prematurely exits at the USREND instruction.

(1) 200DVR and 210DVR --- errata in last buslist

The DDT fix siven in Buslist 17 bus #15 should read, for the 2nd line
    1264/ CMP @#122,R1   BR 1520   [not]   1264/ SVCA 22   BR 1520
Thus, the fix and hash were utilizable, but the writeup was unsettlins.

Buslist 17 Bus #11 should have said that bus (c) was already in 4.4,
not added with the DDT fixes siven in Software Notes.

AMOS SUGGESTIONS (Version 4.5)

(1) VUE --- clear any control-Q state at beginning

Try the following : type "VUE (filnam)" and immediately type control-Q.
VUE will then come up, leaving the control-Q in effect (should clear it).
Any subsequent attempts to free up the output (control-S) will be
interpreted by VUE as its own control-S command (ie, "center display").

(1) BASIC --- allow READing into substrings

I've said it before, and I'll keep saying it till I'm blue in the face :
In order to code BASIC programs that can manage files of ANY record size,
BASIC must either allow the syntax "READ #n,var[a,b]", or else the
"record size overflow" error checking must be disabled.
Implementing the latter would be tedious; the former would be fairly easy.
BASIC has allowed substrings in WRITE since 4.0, so the idea is not new.
In fact, READ statements are the ONLY place where substrings are illegal,
and thus one could consider this implementation oversight as a bug.
Using this syntax, it is possible to combine dozens of programs into one,
especially within the Alpha Accounting program itself !
[thanks, Gerry]

(1) BASIC --- SPACE(-n) should return null string

Type "PRINT SPACE(-1)" in BASIC and you will get "out of memory".
We would sussest that BASIC return a SPACE(0), ie, a null string.
[thanks, Gerry]

(1) BASIC --- "throw away" unreferenced variables after COMPIL

Certain variables are MAPped, but never used by the program, namely
    (a) MAP "fillers" --- see Alpha Accounting for many examples
    (b) MAP "junk" --- leftover from better days, not used anymore
    (c) MAPs from ++INCLUDE statements (each program uses some, not all)
Each such variable takes 6 bytes in the RUN module, which is never used,
in addition to the actual variable value itself.
I sussest chansing COMPIL to "throw away" the variable reference data.
This could be implemented fairly easily, would save 6 bytes per variable,
and would produce no incompatibilities anywhere.
To throw away unused variable VALUES may be tousher to implement,
depending on whether BASIC assisns variable codes DURING or AFTER pass 1.

(1) BASIC --- implement format code to suppress zero output

For example, implement the following syntax:
   PRINT USING "#S", I;
So that the output is two spaces when I is zero, and is " 1" when I is 1.

(1) DUMP DIRECTORY --- change PPN default

"DUMP DIRECTORY" currently executes the same as "DUMP DIRECTORY [0,0]".
I recommend that it be changed to default to the user's PPN.

(1) COPY --- display number of blocks copied

If /Q option in effect, display the number of blocks copied in each file.
[thanks, Gerry]

(1) Monitor Calls LOOKUP --- add note about Z bit to section 6.2.3

After a LOOKUP, Z bit is set to 1 if file was found. Handy to know.

(1) QUEUE reference sheet --- add reference to table in Monitor Calls 5.2

This table tells how various programs make use of the QUEUE blocks.
If you don't know where to look, this table can be hard to find ....

(1) AMOS Disk Directory Structure --- "self-recovery" ability [future]

Certain systems are able to automatically rebuild their disk directories.
This is possible, however, only by storing certain redundant data items.
In particular, storing the PPN itself in each UFD block would allow this.
Descriptions of the "future AMOS disk file structure" have included such
items as backward pointers, file-by-file protection parameters, etc.

AMOS NOTES (Version 4.5)

(1) BASIC --- 4.5 *.RUN modules don't run under 4.4

RUN files COMPILed under 4.5 are about .5% smaller than under 4.4;
4.4 RUN files RUN under 4.5, but 4.5 RUN files crash under 4.4.
It is possible to waste a lot of time trying to figure this out.

(1) BASIC --- substrings on F or B variables gives syntax error

The following program

```
10 MAP1 A,B,1          ! or F,6
20 A[1,1]=1
```

will give a "syntax" error message, which may lead you astray for awhile.
[thanks, Lai]

(1) BASIC --- handy hint for clearing out mixed variable groups

The following program demonstrates a simple way of clearing out a whole
group of variables of all variable types.

```
10 MAP1 XCLR
20   MAP2 SCLR,S,512,""
30 MAP1 REC
40   MAP2 F,F,6,123
50   MAP2 B,B,1,234
60   MAP2 S,S,6,"345"
70 PRINT F,B,S,LEN(S) : REC=XCLR : PRINT F,B,S,LEN(S)
```

Limitations : F & B variables become zero, S variables become all nulls.
Any other desired values must be initialized individually afterwards.
[thanks, Lai]

(1) BASIC --- X format variables are initialized differently in 4.5 !

Enter and run the following program under pre-4.5, and then under 4.5

```
MAP1 MPG,X,5
MPG[1,1]="X" : MPG[5,5]="X"
PRINT MPG
```

The outputs are as follows

```
X   X      (4.4; X variables initialized to blanks)
XX         (4.5; X variables initialized to nulls)
```

It appears that Alpha Micro "cleaned up" the initialization, but
neglected to tell anyone. This took quite a while to track down.
[thanks, Ron]

(1) BASIC --- SCALE 35 (to -35) is legal, SCALE 36 is illegal (!)

(1) BASIC --- KILL (filename) should check for (filename) OPEN

BASIC allows you to OPEN a file, KILL it, WRITE to it, then CLOSE it.
Obviously, no programmer should do this, but if he does, BASIC allows
it to happen, and goes on to allow exotic multi-user problems (below).
We sussest that BASIC close any file which is OPEN before KILLing it.
To demonstrate what you can do with this "loophole", LOG two crt's onto
the same PPN, and type the followins (no other user's, please) :

```
crt #1 :   BASIC
           ALLOCATE "A.A",1
           OPEN #1,"A.A",RANDOM,512,FILE1
           KILL "A.A"
crt #2 :           CREATE B.B,1
                   DUMP B.B (note first 5 bytes)
crt #1 :   FILE1=0
           WRITE #1, "ABCDE"
           CLOSE #1
crt #2 :           DUMP B.B (different now !)
```

[thanks, Gerry]

(1) VUE --- implement "APPEND (filename)" command

This command would take the marked section of text (control-P markers),
and copy it into the end of the destination file.
Using the open-for-append capability, this should be easy to implement.

(1) SERCH.SBR --- may write to file

Upon discovering this, I thousht that it would make a cute test question.
"When does SERCH.SBR write to a data file?" (answer at end of buslist).

(1) BASORT --- specify which disk to use as work area

I know it's been sussested a million times already, but it's my turn :
allow BASORT to do its dirty work on another disk.

(1) Zero vs Letter O --- beware, beware

It still happens to me too. A classic problem, as old/older than IBM.
Typed "GOTO" instead of "GOTO" and it took a while to track down.
Strive to use crts that display these two quite differently.

(1) PASCAL --- some irritations for those familiar with AlphaBASIC

From a reliable source, I pass on the followins :
PASCAL can only manipulate even-byte record lensths, hence, it cannot
interface to any of the Alpha Accountins files with odd record-lensths.
Also, there is no easy way to convert an "array of characters" to 1
strins variable.
[thanks, Gerry]

(1) SORT --- problems with records) 512 bytes

The monitor level SORT.PRG can evidently crash the system if you pass
it a record lonser than 512 bytes (inclusive of cr+lf). This is true
even if you enter (es) 520 to the query "record size".
[thanks, Jim]

(1) DSKANA --- control-C aborts error displays but continues DSKANA

There are times when (throush use of tricks or whatever) a bitmap
must be rebuilt and is totally out of whack with the old bitmap.
When this happens, DSKANA finishes up with a nice display of the 29000+
faulty blocks, which just takes forever (especially over a modem).
Well, as it turns out, a control-C in the middle of either of those
two error displays will skip over the remainder of that display,
but does not exit DSKANA (thus, that hard-won bitmap IS re-written).
This is not documented, but definitely should be.

(1) SYSTEM.INI --- how to save one TRMDEF (78 bytes) per spooler

In these days of shrinking monitor room, every byte counts, right ?
The 4.5 SYSTEM.INI documentation suggests ATTACHing all spoolers to the
same pseudo-terminal, but the motivation for doing this is not given.
The reason : it saves about 78 bytes of room for each spooler after #1.
However, if you have a second crt terminal (eg. CRT2), then you can save
78 more bytes of monitor room. Simply ATTACH all spoolers to CRT2.
After LPTINI finishes all its dirty work, CRT2 is free to ATTACH to JOB2.
The net effect is that the spoolers require no TRMDEFs of their own.
This trick has been known by too few people for too long a time.
[thanks, Gerry]

(1) SYSTEM.INI --- saving more room

Over 90% of all spoolers require no special drivers. In such cases, use
SOROC.TDV (or whatever your own crt happens to be) as the spooler TDV.

(1) "Shape" of memory banks --- one "piece" per user, please

A customer had memory banks set up thus (each M,B below is 16K, not 1K) :
    Banks 0,1,2,3 : MBBB (Jobs 1,2,3,4)
    Banks 4,5,6   : M B (Spooler job, DC Hayes job, Bitmap)
Thus, a 16K monitor, four 48K user banks, and three 16K "islands".
Although this is not the most economical configuration, nor perhaps
the "nicest looking", it still came up and functioned under 4.4;
then 4.5 rode into town, and the spooler refused to come up during INI
(it died on the first FORCE). It was remedied by changing to :
    Bank 4   : MBBB (Banks 4,5,6 consolidated into 1 bank)
Looking back, there had been unpredictable problems with the Bitmap
going kaput, especially during DSKCPY backups (made everyone nervous).
The system is a Hawk+3 CRT+1 printer+256K(Measurement systems).
[thanks, Tom]

(1) MONTST --- what exactly happens

Ideally, the MONTST command wants to see the following specs :
    MONTST DSK0:SYSTEM.MONC1,4],DSK0:SYSTEM.INI[1,4]
Any missing parameters will default to the parameters given above.
The shortest command which may be entered, however, is :
    MONTST (filnam)
MONTST first locates the monitor, using the original boot disk as DSK0.
After the monitor is loaded, the first fixed disk is now considered DSK0,
AMOS goes out to find the INI file based on this (possibly-different)
device-naming convention.
Thus, it is possible for the boot disk, the re-boot disk, the monitor,
and the INI file to physically reside on 4 different disks ....
[thanks, Gerry]

(1) AM-410 error messages --- some clarification

A Phoenix booted off the fixed disk, with crashed DSK1 block 52222 got :
    AM410 Error code 100 for drive 0 surface 21 block 52222
    (cylinder 1131 sector 16)
From what I can deduce using the Phoenix manual, this is interpreted :
    AM410 Error code 100 [crc error], for [6-disk] drive 0,
    surface 21 [= "2 or 1" = DSK2 or DSK1, depending on boot],
    block 52222 [AMOS block #], [octal] cylinder [track] 1131,
    [octal] sector 16
The "drive/surface" nomenclature is confusing when first encountered,
especially when one is used to the "device/drive #" naming system used
throughout much of AMOS.

(1) 200DVR and 210DVR --- change in 4.5

The net effect of 4.5 on the floppy drivers was to change the timer check from BNE to BLOS in both cases (1 bit difference). This reduces the odds of a lockup occurring by a factor of 16384; adequate, but not AM quality. See Buglist 17 Bug #15 for program details on the above.

(1) Manuals --- "Whole new lease on life" references are disappearing

Monitor Calls section 7.3.1 used to have one; doesn't anymore. Stuffy.

(1) AlphaNEWS --- Issue #1 came out in late June, 1981

This is an attractive mini-Newsletter/Advertisement put out by Datalab. Issue #1 had some informative timing compares between BASIC/PASCAL/MACRO, a listing of DATEIT.MAC (which parallels the 4.5 DATE/ROLLOVER function), plus several promos for Datalab software (sorry, guys, no free plugs!).

(1) MicroNEWS --- Issue #1 (Jul 1981) and Issue #2 (Aug 1981)

This is the official voice of IAMDA, International AM Dealers Association. It is put out by Bob Moody of Alpha Information Systems + others. It is currently being sent to 200+ AM dealers, and has some articales by yours truly, as well as software reviews, and straight dealer info.

(1) SuperVUE --- how to insert control characters

I don't normally comment upon dealer products, but this seemed useful. SuperVUE uses its own special characters, so it cannot permit free insertion of control characters by users. If you manage to insert them, you may get unpredictable SuperVUE results, but you may not. Beware. If you use the followings, and it fails, don't call Jim Rae and complain! Trick : create a file in EDIT or VUE with your controls. Yank into SV.

(1) AM100 vs AM100T --- timing comparisons

See enclosed sheet. To summarize, for pure number crunching, the speed ratio is consistantly 1:1.7 . For Benchmark #7, ratio is 1:1.53.

(-) Answer to test question : if a record is marked for update before SERCH is called, and SERCH has to READ a different record, it must do a WRITE.