alpha micro

Software Notes

V4.4b(1)

September 1980
Volume I, Number 4

Contents of this Issue

INTRODUCTION

This is the Software Notes, the Alpha Micro software newsletter written for our Dealer and OEM network. If you have software information or hints that you would like to share with others, or if you have software questions you would like answered, please write to:

Editor, Alpha Micro Software Notes
Software Group
P.O. Box 18347
Irvine, California 92713

POLICIES AND PROCEDURES

Ordering Additional "Software Notes"

We realize that many dealers have several people within their organization who might benefit from a copy of the Software Notes. Also, many of the sample programs and software hints found in the Software Notes will continue to be relevant for future software releases.

Therefore, we have decided to make both back issues and multiple copies of future issues of the Software Notes available to dealers on the Software Notes mailing list. Please use a copy of the subscription form at the back of this issue for ordering back issues and multiple copies of the Software Notes.

If you would like to correct or change the mailing address we use in mailing this newsletter to you, please use the subscription form for that purpose as well.

While we're discussing plans for the Software Notes, we'd like to mention that we plan to create an annual index covering each year's issues of Software Notes. The first index will be released January, 1981.

AMOS Version 4.4b

Just a reminder-- the recently released 4.4b System Disk contains all of Software Release 4.4a plus:

1. The patches listed in the August issue (Volume I, #3) of Software Notes.

2. All the files that make up the AlphaPASCAL Version 2.0 programming system, including: the compiler; the linker; the standard library; and various Include-files.

NOTE: You must order the new AlphaPASCAL manual separately from the Release Disk. Order the AlphaPASCAL User's Manual, (part number DWM-00100-08, Revision B00).

3. A change to the monitor so that it displays "4.4b" when you run SYSTAT or SYSTEM. The hash totals for the various versions of the 4.4b SYSTEM.MON are:

Phoenix Monitor	144-036-454-061
Hawk Monitor	065-512-215-410
DDA Monitor	014-264-144-006
Persci AMS Monitor	360-364-162-150
Wangco AMS Monitor	507-741-714-373
Persci STD Monitor	505-771-702-217
Wangco STD Monitor	053-532-627-575

SOFTWARE HINTS

AlphaAccounting Subroutines

It has come to our attention that some of our users are using the wrong versions of the AlphaAccounting assembly language subroutines. It is very important that you use the versions of the subroutines that were included with the AlphaAccounting Version 1.3 Release Disk. If you are experiencing problems, chances are that you have outdated copies of the subroutines.

To make sure that you have the correct subroutines, use the DIR/H command to generate hash totals for your subroutines, and check those hash totals with the ones listed below:

ACCEPT.SBR	730-113-040-142	ANYCN.SBR	555-535-613-655
DSPLY.SBR	101-464-270-344	ECHO.SBR	127-450-177-722
INPUT.SBR	172-136-520-351	MESAG.SBR	611-771-575-453
MMENU.SBR	012-262-414-271	MOUNT.SBR	600-277-471-717
NOECHO.SBR	672-253-376-122	PGMND.SBR	241-456-233-714
PRINT.SBR	155-760-564-361	PRIV.SBR	564-023-642-307
RDATE.SBR	333-642-122-060	RENAME.SBR	015-750-755-545
SCREEN.SBR	241-454-412-157	SERCH.SBR	216-324-525-723
SRCH2.SBR	546-267-440-237	STALL.SBR	140-276-665-705
STENO.SBR	161-542-774-377	STRIP.SBR	767-505-536-145
TMENU.SBR	420-213-240-234	TMEN2.SBR	131-776-432-017
WAIT.SBR	156-306-706-245		

If the hash totals do not match, you do not have the current versions of the subroutines.

AlphaBASIC

1. AlphaBASIC Program Labels

Although not discussed in the AlphaBASIC User's Manual, (DWM-00100-01), you can use reserved words as program labels when your program is in the normal EXPAND mode. While intended as a convenience, this feature has caused some confusion when a user thinks that he or she is specifying a statement and AlphaBASIC reads the keyword as a label. For example, consider the case where you want to put several null PRINT statements on multi-statement lines:

```
10 PRINT:PRINT:PRINT
20 PRINT:PRINT:PRINT
```

If you attempt to compile the program lines above, BASIC reports a "Duplicate label" error message. While you intended the colon as a statement separator, BASIC read it as a label terminator. If you wish to put several null PRINT statements on a single line, place blanks between the colons and the reserved words:

```
10 PRINT : PRINT : PRINT
```

This ensures that BASIC reads the colons as statement separators, not as label terminators.

also watch out for NEXT:NEXT

2. BASIC Program to Calculate Pi (3.14159...)

Our thanks to the user who sent in the following suggestion for calculating Pi to 11 significant digits:

```
10 SIGNIFICANCE 11
20 PI = 2 * ASN(1)
```

3. Error-Reporting Bug

A particularly elusive error-reporting problem has been found with COMPIL. The symptoms are: a program compiles without apparent error but, when run, the program arbitrarily restarts itself at the beginning when it reaches a GOTO statement somewhere in the program.

It turns out that these symptoms result from an actual syntax error in the program that was not reported by COMPIL; this error causes COMPIL's GOTO handling to go awry. For information on a patch to COMPIL that enables it to report such errors, see the section below titled "Software Change Notices."

AlphaPASCAL Version 2.0

1. Undocumented Functions and Procedures

Because the AlphaPASCAL User's Manual was sent to the printers before the final touches were made to AlphaPASCAL Version 2.0, several functions and procedures are undocumented. These functions and procedures are VAL, STR, RANDOMIZE, and RND. The VAL and STR functions allow you to convert data between string and numeric format, and vice versa. This is necessary when you want to use both numeric and string functions on the same piece of data. RANDOMIZE and RND allow you to generate random numbers.

VAL -- The VAL function accepts a STRING argument and returns a REAL number. It takes the form:

```
VAL(source-string);
```

For example:

```
PROGRAM TestVAL; { Convert string input to REAL form }

VAR     Salary : STRING;
        Number : REAL;
BEGIN
  WRITE('Enter your yearly salary: '); READLN(Salary);
    { Use string function LENGTH on Salary }
  IF LENGTH(Salary) < 5 THEN WRITELN('You need a raise!');
  Number := VAL(Salary); { Convert string to number }
  WRITELN('If they keep 37% of your salary, you''ll only see');
  Number -= (Number * 0.37);
  WRITELN(Number,' dollars of your salary!')
END.
```

STR -- The STR function accepts a REAL or INTEGER number and returns a STRING. It takes the form:

```
STR(Number);
```

For example:

```
PROGRAM TestSTR; { Convert REAL number to STRING }

VAR      Price : REAL; Target : STRING;

BEGIN
   WRITE('Enter price of object $');
   READLN(Price);
   WRITELN('Sales tax is:',Price * 0.06); { Perform numeric operation }
   Target := STR(Price); { Convert to string }
   { Use string function, POS, on data to search for decimal point. }
   IF POS('.',Target) = 0 THEN WRITELN('The price is in whole dollars.')
END.
```

You may also use STR to format a converted number by specifying two optional INTEGER arguments:

```
        STR(Number,X,Y);
or:
        STR(Number,X);
```

where X specifies a minimum field width, and Y specifies the number of digits to write after the decimal point. (If "Number" is INTEGER, you may not specify Y.) These two variations of STR perform formatting in exactly the same way as WRITE and WRITELN, except that they do not generate a leading space for positive numbers. (See Section 10.1.5.5, "Formatting Output," in the AlphaPASCAL User's Manual, (DWM-00100-08, Revision B00), for information on WRITE and WRITELN.) For example, given the REAL data 123.44:

```
        WRITELN(STR(123.44,10,4));
```

returns the string:

```
  123.4400
```

where the number is right-justified in a field of ten blanks, and four digits are written to the right of the decimal point. X specifies the minimum field in which to print the number-- if the number is larger than the specified field, AlphaPASCAL does not truncate the number, but prints the number using the necessary number of digit positions.

RANDOMIZE and RND -- The RND function returns a random REAL number between 0 and 1. (It acts exactly like the RND(0) function of AlphaBASIC.) It takes the form:

```
        RND;
```

(with no arguments). To randomize the starting seed of the RND function, use the RANDOMIZE procedure:

```
        RANDOMIZE;
```

(with no arguments). For example:

```
        PROGRAM TestRND;
        { Generate 20 random integers between 1 and 10 }
        VAR     I : INTEGER;
        BEGIN
          RANDOMIZE;
          WRITELN('Random numbers between 1 and 10:');
          FOR I := 1 TO 20 DO
            BEGIN
              WRITELN(TRUNC((RND*10)+1))
            END
        END.
```

2. PASCAL Program to Pad Number with Leading Zeros

Below is a useful procedure to pad a number with leading zeros along with a sample program that makes use of it:

```
     PROGRAM FormatTest;

     VAR       S : STRING;
               { The procedure call Format(String,Left,Right,Number) formats
                 number with Left zero-filled digits before the decimal point
                 and Right zero-filled digits after the decimal point.  A
                 trailing space or minus sign is added to indicate the sign of
                 the number.  Illegal arguments generate an error to ERRORTRAP.}

     PROCEDURE Format(VAR X : STRING; Left,Right : INTEGER; Num : REAL);
        VAR  Pow : REAL;
        BEGIN { Procedure Format }
           IF Left > 11 OR Left <= 0 THEN ERROR(1);
           Pow := PWROFTEN(Left);
           IF ABS(Num) >= Pow THEN ERROR(1) { Value range error };
           X := STR(Pow + ABS(Num),0,Right);
             { Force leading zeros by adding power of ten and converting
               to STRING. }
           DELETE(X,1,1);     { Remove leading 1 }
           X := IF Num < 0 THEN CONCAT(X,'-') ELSE CONCAT(X,' ');
        END { Format };

     BEGIN { Main Program }
        Format(S,5,2,-12.7); { Return answer in S }
        WRITELN('Format(5,2,-12.7) = ',S);
        WRITELN('Result should be 00012.70-');
     END { Main Program }.
```

Miscellaneous

CRT410 -- If you interrupt CRT410 via a Control-C during its initial prompt sequence, CRT410 writes a BADBLK.SYS file to the surface that contains a list of zero bad blocks. Also, if interrupted at this point, CRT410 does not generate a bad hash total for the BADLBK.SYS file. Therefore, the monitor has no way of knowing that the certification was incomplete. If you attempt to use such a surface, it is probable that the system will attempt to write to blocks that may be bad, and that you will lose your data.

If you want to interrupt a CRT410 certification, make sure that you do so after the certification has begun so that the system will be able to identify the surface as incompletely certified. If it is absolutely necessary to interrupt CRT410 while it is still prompting you for information, make sure that you immediately certify the disk before anyone tries to use it.

TXTFMT -- Although both commands work correctly separately, a problem occurs with TXTFMT when you try to use the /FOOTER and the /ODD PAGE commands together-- TXTFMT does not paginate correctly. This problem will be fixed in Release 4.5. Until then, if you are going to use both /ODD PAGE and /FOOTER commands, we suggest that you work around the problem by simulating the /ODD PAGE command by including the proper number of /PAGE commands.

DSKANA -- If you use the SET HEX command to force hexadecimal output, the DSKANA program incorrectly displays project-programmer numbers in hex. This problem will be fixed for Release 4.5.

AlphaFIX -- AlphaFIX, the screen-oriented assembly language program debugger, contains a display problem. If you use FIX in disassembly mode and it encounters a TJMP instruction, FIX disassembles all code after the TJMP as OFFSETs.

To correct the screen display, use the line-feed command to scroll the TJMP instruction off the top of the screen; then type two Escapes to enter and immediately exit Command mode. The screen display will now be correct.

MACRO -- MACRO does not recognize the indirect addressing mode for floating point instructions. This problem will be corrected in Release 4.5. You can work around this limitation until then by hand-assembling the instruction to its octal or hexadecimal equivalent.

SRCCOM -- The SRCCOM program may crash the system if you use it to compare very large files. This problem will be fixed in Release 4.5; until then, you can temporarily get around the problem by increasing the size of your memory partition before using SRCCOM.

SYSTAT -- The system may crash if you load SYSTAT into memory and then execute it while the System Disk is not ready. This is especially likely to happen to floppy disk users, since they often load programs into memory from the System Disk and then remove the disk from the drive. This problem will be fixed in Release 4.5.

DOCUMENTATION ADJUSTMENTS

AlphaFIX

Page 2-5 of the AlphaFIX User's Manual, (DWM-00100-69), indicates that LOCK is equivalent to the IEN machine instruction; it is actually equivalent to the IDS instruction.

ISAM

IMPORTANT NOTE FOR ASSEMBLY LANGUAGE PROGRAMMERS: The current ISAM System User's Guide, (DWM-00100-06, Revision A01), contains the following errors in Chapter 5, "Using ISAM from the Assembly Language Level," pages 5-7 and 5-9:

Page 5-7:

```
Was:          CALL    .IRLRD
Should be:    CALL    .IRLRD(Rn)

Was:          CALL    .IRLWT
Should be:    CALL    .IRLWT(Rn)
```

Page 5-9:

```
Was:          CALL    .IDELK
Should be:    CALL    .IDELK(Rn)
```

NOTE: It is very important that you make this correction to your ISAM manual!

VUE HELP Command

The current AlphaVUE User's Manual contains a description of the use of the HELP command in the INI.VUE file to disable/enable the display of the VUE command menus. The manual also discusses using the HELP? command in Command mode to disable/enable menu display.

Another, undocumented feature of the HELP command exists. When in Command mode, if HELP has been set to TRUE in the INI.VUE file, typing HELP (without the question mark) followed by a RETURN causes VUE to display a list of the topics for which menus exist. For example:

```
>HELP RET

Help is available for:
SCREEN   COMMAND   TXTFMT   BASIC   BASICIO   TCRT   DDB
```

You can then see the menu for a specific topic by typing HELP, the topic name, and a RETURN. For example:

```
>HELP TXTFMT RET
```

displays a menu of TXTFMT commands.

(NOTE: To use the HELP? or HELP command in Command mode, the HELP command must have been previously set to TRUE in the INI.VUE file.)

SOFTWARE CHANGE NOTICES

The patches below assume that you have a monitor version 4.4a or later. The patches below will work on a 4.4 system, but we feel that it does not make much sense to perform the patches in this case, since you will have to perform them again as soon as you update to 4.4a or 4.4b. Therefore, we recommend that you not perform the patches unless you are operating under version 4.4a or later.

NOTE: If you are going to make the patches below, it is a good idea to make all of them, even if you do not use some of the programs affected; otherwise, you will have future support problems because you will have a non-standard mix of software.

Because the monitor patch at the end of this issue identifies your system software as containing all of the patches in this issue, do not make the SYSTEM.MON patch until you perform all of the other patches.

COMPIL.PRG

As we mentioned above in the "Software Hints" section, a problem exists with COMPIL's error trapping for certain kinds of syntax errors. Although the problem rarely occurs, it has a serious effect on your compiled program.

The symptom of the problem is that COMPIL compiles the program (reporting no errors), but when the program is executed, the program arbitrarily restarts itself at the beginning when it reaches some GOTO in the program.

We have discovered that when COMPIL is executing an error processing routine in the first pass of the compiler, an internal stack used for storing unresolved transfers of control is also used as a temporary stack space for processing the error itself. When COMPIL tries to restore the stack on completion of the error checking, it loses the error along with the remainder of the stack. Consequently, when RUN encounters the unresolved GOTOs and CALLs, it doesn't know where to go, so it simply restarts the program at the beginning. Also, although the error is found on the first pass, COMPIL does not report it.

This problem will be fixed in Release 4.5. Until then, we have a patch that will enable you to detect such errors during compilation so that you can correct your problem before running the program. Since the error in your program was, in fact, discovered by COMPIL but was simply unreported, we can force COMPIL to report the error in Phase 1 of the compilation. This patch allows you to find the syntax error if you use the /T option of COMPIL. (Your clue that such an error exists is the symptom of a program restarting from the beginning on its own.) Following the patch below is a sample compilation (using the /T option) of a program that contains the type of error that causes the problem discussed above.

Although the particular syntax problem that alerted us to this problem involved an invalid subscript specification (too many or not enough parentheses), other types of syntax errors may cause the problem. To help us in tracking down other problem areas in COMPIL's error handling routines, if you write a BASIC program that exhibits the behavior described above, we would like to request that you send us a listing of the program's compilation (using the /T option) with an SPR.

```
.LOG SYS:(RET)
Logged into DSK0:[1,4]
.DIR COMPIL.PRG/H(RET)
COMPIL  PRG 20   745-533-402-115
.DDT COMPIL.PRG(RET)
PROGRAM BASE IS xxxxxx
PROGRAM SIZE IS 23456
14/     LEA R3,23454  LEA R3,23534(RET)

11040/  MOV R5,12(SP) JMP 23454(RET)

23454/  0             MOV R5,12(SP)↓
23460/                TST 502(R0)↓
23464/                BNE 23472↓
23466/                JMP 11016↓
23472/                SVCA 6↓
23474/                SUB @-(R0),@-(R2)↓
23476/                SUB @-(R0),@-(R2)↓
23500/                SUB @-(R0),@-(R2)↓
23502/                SUB @-(R0),@-(R2)↓
23504/                SUB R0,-(R0)↓
23506/                0↓
23510/                MOVB -(R1),R3↓
23512/                CALL R1,23526↓
23516/                0↓
23520/                SVCA 10↓
23522/                JMP 11016↓
23526/                SAVE↓
23530/                JMP 10744↓
23534/                0(RET)
^C
.DIR MEM:COMPIL.PRG/H(RET)
COMPIL PRG 10078 677-447-465-767        MEM:
.SAVE COMPIL.PRG(RET)
ERASE COMPIL.PRG,       SAVE COMPIL.PRG
.DEL COMPIL.PRG(RET)
COMPIL.PRG
```

The patched version of COMPIL allows you to use the /T option to detect the type of syntax errors that cause a program to restart. (You will not see the error if you don't use the /T option.) Below is a sample program that contains such an error along with a sample compilation of that program (using the /T option).

Notice that lines 80 and 110 both contain errors that COMPIL normally reports; line 120, however, contains an error that an un-patched version of COMPIL will not report.

```
10      ! This program has lots of errors to demonstrate the
20      ! /T option with COMPIL
30
40      MAP1 A,B,1                      ! Binary Variable
50      MAP1 R(100),F,6                 ! Floating Point array
60
70      INPUT A
80      A=A*/2                          ! <==Error
90
100     FOR I=1 TO 100
110      R(I,4)=A//5                    ! <==Error
120      R(I)=R(I&)+R(I,5)-4            ! <==Error
130     NEXT I
140     CALL SUB'ROUT
150     END
160 SUB'ROUT:
170     PRINT "THIS IS THE SUBROUTINE"
180     RETURN
```

Now we will compile the program using a patched COMPIL and the /T option. Notice that COMPIL now reports the illegal subscript error in line 120 when you use the /T trace option, but that the error still is not reported during Phase 2. Also notice that there is more than one error in lines 110 and 120-- BASIC or COMPIL by design finds only the first error in a program line.

```
.COMPIL TEST/T(RET)
Phase 1 - Initial work memory is 2320 bytes
10      ! This program has lots of errors to demonstrate the
20      ! /T option with COMPIL
30
40      MAP1 A,B,1                          ! Binary Variable
50      MAP1 R(100),F,6                     ! Floating Point array
60
70      INPUT A
80      A=A*/2                              ! <==Error
********  Syntax error
90
100     FOR I=1 TO 100
110       R(I,4)=A//5                       ! <==Error
********  Wrong number of subscripts
120       R(I)=R(I&)+R(I,5)-4                                   ! <==Error
********  Illegal subscript
130     NEXT I
140     CALL SUB'ROUT
150     END
160 SUB'ROUT:
170     PRINT "THIS IS THE SUBROUTINE"
180     RETURN

Phase 2 - Adjust object file and process errors
Syntax error - 80      A=A*/2                   ! <== Error
Wrong number of subscripts - 110      R(I,4)=A//5      ! <==Error
Memory usage:
        Total work space - 2464 bytes
        Label symbol tree - 10 bytes
        Variable symbol tree - 52 bytes
        Data statment pool - 0 bytes
        Variable indexing area - 28 bytes
        Compiler work stack - 14 bytes
        Excess available memory - 7972 bytes
End of compilation
.
```

ACTIV.TDV

Users of the ActIV terminal will experience problems when using the VUE block marker command, ^P. The symptom is that the marked block displays in alternate lines of light and dark intensity. The problem occurs because VUE assumes that a change in the intensity of the ActIV terminal display can be specifically set (as is the case with most terminals). However, the intensity function of the ActIV terminal is a "toggle"; therefore, sending the "set intensity" codes to the terminal driver merely causes the terminal display to alternate between light and dark lines. The cure for this problem is to disable the reduced intensity function of the ActIV terminal driver. The patched terminal driver still allows VUE to accept a ^P command to mark a block, but the marked block displays in normal intensity.

Below we give both the patch for the assembled and linked program file and the changes to make in the original driver source file. For either change, the hash totals should be:

```
Old Hash Total:      317-452-524-125
New Hash Total:      425-435-064-136
                              452
```

NOTE: These changes to the ActIV driver disable the intensity TCRT calls (-1,11) and (-1,12) If your BASIC, PASCAL, or assembly language programs need to make use of these calls, do not make the changes below.

The following lines show the changes you should make to the source file for the ActIV driver, DSK0:ACTIV.MAC[10,2].

```
Was:           C11:   BYTE   14.           ; Change intensity
Should be:     C11:   BYTE   0.            ; Disable intensity

Was:           C12:   BYTE   14.           ; Change intensity
Should be:     C12:   BYTE   0.            ; Disable intensity
```

After changing the source file, you will need to re-assemble and link it. To do this, enter:

```
.MACRO ACTIV(RET)
```

Now you see something like this:

```
PHASE 1: MEMORY USED - nnnn WORDS
PHASE 2: OBJECT FILE FINISHED
PHASE 4: PROGRAM FILE FINISHED
```

Now, rename the .PRG file you have just created to the terminal driver extension, .TDV. Then log into the System Driver Library account, and copy the new driver into the account.

```
.RENAME/DELETE .TDV=ACTIV.PRG(RET)
ACTIV.PRG renamed to ACTIV.TDV
Total of 1 file renamed

.LOG [1,6](RET)
Transferred from DSK0:[10,2] to DSK0:[1,6]
Ersatz name is DVR:

.COPY = ACTIV.TDV[10,2](RET)
ACTIV.TDV[10,2] to ACTIV.TDV
Total of 1 file copied
```

If you do not have ACTIV.MAC[10,2], and want to make the changes above, you can make the following patch to the terminal driver, ACTIV.TDV[1,6]:

```
.LOG 1,6(RET)
Logged into DSK0:[1,6]
Ersatz name is DVR:
.DIR ACTIV.TDV/H(RET)
ACTIV  TDV     317-452-524-125          DSK0:[1,6]
.DDT ACTIV.TDV(RET)
PROGRAM BASE IS nnnn
PROGRAM SIZE IS 372
360/    MBWU   R1,SP   0(RET)
^C
.DIR MEM:ACTIV.TDV/H(RET)
ACTIV  TDV     425-435-064-136          MEM:
.SAVE ACTIV.TDV(RET)
ERASE ACTIV.TDV        SAVE ACTIV.TDV
```

FLOCK.SBR

A problem has been reported for a system with three or more terminals using FLOCK, the BASIC Multi-user File Locking Subroutine. The patch below corrects this and other problems (For information on using FLOCK, see the "BASIC Programmer's Information" section of the AM-100 documentation packet.)

If you currently have a version of FLOCK with the hash total

    647-745-351-666

perform this patch:

```
.LOG BAS:(RET)
Logged into BAS:
.DIR FLOCK.SBR/H(RET)
FLOCK  SBR     647-745-351-666          DSK0:[7,6]
.DDT FLOCK.SBR(RET)
PROGRAM BASE IS nnnnn
PROGRAM SIZE IS 2066
1676/   MOV R2,10(R0)         JMP 2174(RET)

2174/                         CLR 6(R0)↓
2200/                         MOV R2,10(R0)↓
2204/                         JMP 1702(RET)
^C
.DIR MEM:FLOCK.SBR/H(RET)
FLOCK  SBR     600-131-601-745          MEM:
.SAVE FLOCK.SBR(RET)
ERASE  FLOCK.SBR,      SAVE FLOCK.SBR
.DEL FLOCK.SBR(RET)
FLOCK.SBR
```

If your version of FLOCK.SBR has the following hash total:

    446-075-217-674

perform this patch:

```
.LOG BAS:(RET)
Logged into BAS:
.DIR FLOCK.SBR/H(RET)
FLOCK   SBR     446-075-217-674          DSK0:[7,6]
.DDT FLOCK.SBR(RET)
PROGRAM BASE IS nnnnn
PROGRAM SIZE IS 2066
130/    XCH R1,@SP              JMP 2066(RET)

176/    SVCB 0                  JMP 2110(RET)

322/    SVCB 0                  JMP 2136(RET)

1574/   BIS #1,10(R3)           BIS #2,10(R3)(RET)

1676/   MOV R2,10(R0)           JMP 2174(RET)

2066/                           LEA R2,74(R5)↓
2072/                           MOV R2,2(R5)↓
2076/                           XCH R1,@SP↓
2100/                           SUB 156(R0),R1↓
2104/                           JMP 136↓
2110/                           BIC #1400,0(R5)↓
2116/                           SVCB 0↓
2120/                           XOR 0(R5),R0↓
2124/                           BIS #1400,0(R5)↓
2132/                           JMP 206↓
2136/                           BIC #1400,0(R2)↓
2144/                           SVCB 0↓
2146/                           SUB 0(R2),R0↓
2152/                           SVCB 0↓
2154/                           ADD 0(R2),#4↓
2162/                           BIS #1400,0(R2)↓
2170/                           JMP 340↓
2174/                           CLR 6(R0)↓
2200/                           MOV R2,10(R0)↓
2204/                           JMP 1702(RET)
^C
.DIR MEM:FLOCK.SBR/H(RET)
FLOCK   SBR     600-131-601-745          MEM:
.SAVE FLOCK.SBR(RET)
ERASE  FLOCK.SBR,      SAVE FLOCK.SBR
.DEL FLOCK.SBR(RET)
FLOCK.SBR
```

SYSTEM.MON

This patch changes the current monitor revision code from "4.4a" to "4.4a(1)" (or from "4.4b" to "4.4b(1)"). That is, it leaves the current monitor level code unchanged, but appends a "(1)" to the end of it. The purpose of the patch is so that your monitor can be identified in the future as containing all of the patches in this issue of the Software Notes; therefore, do not make this patch until you have performed all of the patches above.

To check that the patch has been done correctly-- after you have made the patch, reboot the system and use the SYSTEM command or the S[illegible]TAT command; either command will display the new monitor version.

IMPORTANT NOTE: This patch will not work with a monitor version 4.4 (as opposed to 4.4a or 4.4b). (It won't hurt anything, but it won't cause the "(1)" after the version code to be displayed.) This is because we assume that if you are using a 4.4 version of the software, you will want to update to version 4.4a or 4.4b before making any of the patches in this issue.

```
.LOG SYS:(RET)
Logged into SYS:
.DDT SYSTEM.MON(RET)
PROGRAM BASE IS xxxxx
PROGRAM SIZE IS 32722
14/     0      30450(RET)
16/     0      51(RET)
^C
.SAVE SYSTEM.MON(RET)
ERASE SYSTEM.MON,       SAVE SYSTEM.MON
```

Now reboot and use the SYSTEM or SYSTAT command. For example:

```
.SYSTEM(RET)
The following programs are allocated in system memory:
        STD    DVR
        AMS    DVR
        TRM    DVR
Total resident monitor size is 15466 bytes
Monitor version is 4.4b(1)
```

NOTE: We did not include before and after hash totals in the patch above. This is because the purpose of a hash total is to verify that you have performed a patch correctly. You can more conveniently check this by using the SYSTEM or SYSTAT command as shown above after rebooting the system.