alpha micro

Software Notes

V4.4a

August 1980
Volume I, Number 3

Contents of this Issue

ALPHA MICROSYSTEMS
17881 Sky Park North
Irvine, CA 92714

INTRODUCTION

This is the third issue of Software Notes, the Alpha Micro software newsletter written for our Dealer and OEM network. The last issue (June, 1980) mentioned the upcoming availability of AMOS version 4.4a. This version is now available and is being shipped directly to end-users who are on the direct-update service. The 4.4a system disk contains:

1. The five patches listed in the last issue of Software Notes.

2. The first patch to SYSTEM.MON included on page 12 of this issue.

3. A copy of FIX.PRG in [1,4]

4. A copy of MENDEF.MAC in [7,7].

Dealers may order a 4.4a disk from our Sales Order Department, or pick up the software from the next system delivered to you.

FUTURE PRODUCT ANNOUNCEMENTS

AlphaPASCAL

Alpha Micro is proud to announce the availability of AlphaPASCAL Version 2.0, a completely new version of the popular PASCAL language for the Alpha Micro computer system. As mentioned in previous issues of Software Notes, this new version supports complete AMOS file compatability, assembly language subroutines, separate compilation of modules, and many more features to fully integrate it into the Alpha Micro computer system.

The Sales Order Department will start accepting orders for AlphaPASCAL on August 18, 1980. (Please don't try to order it before then, as we will be unable to process your order.) AlphaPASCAL is being made available at this time on a special order basis; it will be distributed normally on the next scheduled release of AMOS (Version 4.5). Until then, you must specifically order a copy. AlphaPASCAL disks are available in the following formats:

STD Format Floppy Disk (LISP/PASCAL Disk)	$ 50.00
5 Mb Disk Cartridge (contains PASCAL only)	$125.00

Both formats are accompanied by one copy of the AlphaPASCAL manual (DWM-00100-08). Additional copies may be ordered at the same time.

NEW FEATURES OF THE 4.4 RELEASE

AlphaVUE 2.4

The 2.4 version of AlphaVUE contains many new features. For users already familiar with AlphaVUE, this section summarizes the changes. For more complete details on any feature of VUE, consult the new _AlphaVUE User's Manual_ (DWM-00100-15, Revision B00). This updated manual now contains a section for beginners.

The first change to note is that the 2.4 version of VUE is approximately 4000 bytes longer than earlier releases. If you are running VUE in system memory, you may have to allocate more memory space. Other changes to VUE are as follows:

New Screen-Mode Commands

Control-N (Go to end of line): Moves cursor immediately to the end of the current line.

Control-RUBOUT (Erase line): Erases all characters on the current line and repositions the cursor at the beginning of the line. Does not delete the line space itself. (Acts as a Control-U followed by Control-Y.)

Control-P (Set block marker): Allows you to mark off blocks of text for deletion, for movement to other locations in the file, or for transferrence to a new file. Pressing Control-P while the cursor is at the beginning of the block and again while it is stationed at the end causes the block to be displayed in reduced intensity (on terminals with that capability). If you wish to move, copy, or delete only one line of text, you must mark the line by pressing Control-P twice while the cursor is stationed somewhere on that line. Please note that the old format of block marking ([*, *]) is still supported. The following commands issued in command mode manipulate the marked block:

DELETE--Deletes the block of text and moves subsequent lines up to fill in the space.

MOVE--Moves the block to wherever you station the cursor, and deletes the block from the original position. Fills in the space with subsequent text.

COPY--Copies the block to wherever you station the cursor, without deleting it from the original position. Leaves the original block marked.

UNYANK {FILENAME.EXT}--Writes the block to a new file and assigns it the name specified, without deleting the block from the original file. Leaves the original block marked.

CLEAR--Clears the text of all block markers.

Control-G (Absolute character insert): Allows you to insert most control characters into a file using VUE. For instructions, refer to the AlphaVUE User's Manual.

New Mode Settings

Wrap Mode: When Wrap Mode is active, you need not enter carriage returns. When you reach the end of the line, VUE automatically moves the cursor, along with any unfinished word, to the beginning of the next line.

Entry Mode: Has been upgraded for BASIC programmers. It now provides automatic line numbering when needed.

New Command Mode Features

There are now over 40 commands available in VUE command mode. The new commands and adjustments to former commands are outlined below. (The input processing has been upgraded so that it is no longer possible to edit the prompt character.)

SAVE: Saves the current memory image as does the FINISH command, but does not leave VUE. It is useful for periodically backing up your work without having to leave VUE. An important note, however, is that SAVE only works for files that fit entirely into memory; if the file doesn't fit into memory, nothing is saved.

YANK and UNYANK: These commands have been expanded to allow you to merge and split files. Using Control-P as a block marker, you may now write part of the current file to a new one (UNYANK followed by the name of the new file), or copy a file to any specified place in the current one (place cursor, then type YANK followed by the file to be copied in). This capability is especially useful for creating custom documents from standard paragraphs.

In addition, the system no longer leaves stray blank lines when you use YANK and UNYANK to move text in and out of memory.

DIR: The DIR command causes the system to display a list of all files in the current account with the same extension as the file you are editing. This command is helpful when you need to verify the name of a file to be YANKed into the current file, without leaving VUE. Also, since the UNYANK command, when followed by a file name, writes over any file with the same name in that account, DIR allows you to check the directory first to prevent inadvertent deletion of data. As an added convenience, the DIR command supports full wildcarding.

ERASE: Allows you to delete any file in the current account.

NEXT: Similar to the SEARCH command, but starts the search at the current cursor location.

WHOLE: Causes a search of the entire file, even if the file does not fit into memory.

REPLACE and GLOBAL: It is now possible to escape (ESC or Control-C) while entering the replacement string. Also, both ESC and Control-C, as well as Q, terminate the replacement operation.

CENTER and FORMAT: The CENTER command now centers the current line in relation to the WIDTH specification, instead of centering it in relation to the screen width.

In addition, the FORMAT command formats according to the line length specified by the WIDTH command.

WIDTH: When followed by a number, sets the formatting width of the text to the number specified. If no number is specified, the width is reset to the screen width. This command governs the effect of the CENTER and FORMAT commands.

SPLIT and UNSPLIT: The SPLIT command splits the memory image at the current cursor location. This capability speeds up the screen editing process of long files. The only part of the file that can be edited is the text prior to the cursor position before the split.

UNSPLIT rejoins the text that was separated with the SPLIT command. The UNSPLIT command is executed automatically before any disk transfer operation, such as FINISH, GO, SAVE, YANK, or UNYANK.

TEXT: This is a parameter command, which tells VUE to consider the current file as a text file, regardless of the extension.

SBLK: This is a parameter command, directing that any subsequent SEARCH command search only for strings within marked blocks of text. This is especially useful when doing a GLOBAL replace. Only strings within marked blocks are replaced.

DELTA: This parameter command, followed by a number, sets the automatic line-number increment value in Entry Mode to be used when creating .BAS files.

New Initialization Features

The VUE initialization processing has been completely revised internally. It is now possible to leave VUE while it is initially reading in a file, by pressing Control-C.

VUE now operates correctly with INI.VUE in memory, and finds the menu on the System Disk properly. If desired, a :T may be placed at the front of the INI.VUE file so that the file is displayed as it is processed, the same way .CMD and .DO files can be displayed.

Improved Hardware Support

VUE now supports a new terminal driver format which allows nonstandard screen sizes and automatically determines what features a terminal has. Details on these features will be provided in the next issue of Software Notes.

QUESTIONS AND ANSWERS

We have had some inquiries regarding access to the SET functions through assembly language. These functions are enabled or disabled by status bits stored within the JOBTYP word of the Job Control Block (JCB). The names and meanings of these bits are defined in the AMOS Monitor Calls Manual (DWM-00100-42), on pages 2-6. You can modify these status bits by doing a JOBGET monitor call to get the current contents of JOBTYP, changing the contents, and placing the modified word back into JOBTYP through use of the JOBSET monitor call. Both JOBGET and JOBSET are documented on pages 2-3 of the Monitor Calls Manual.

EXAMPLE:

To set the output radix to hex, you could do the following:

```
JOBGET  R1,JOBTYP              ; get job type word
BIS     #J.HEX,R1              ; turn on hex mode
JOBSET  R1,JOBTYP              ; store new job type
```

Likewise, to set octal mode, you would merely clear the J.HEX bit via a BIC instruction.

DOCUMENTATION ADJUSTMENTS

The Change Page Packet #1 for the AMOS System Commands Reference Manual (DWM-00100-49) indicates that a reference sheet for the system command, MACRO, should have been included. However, MACRO did not change for release 4.4, so a new refence sheet was not necessary.

Page 9 of the last issue of Software Notes (June, 1980) contains a patch for DUMP.PRG, which requires a correction. The 520 in line 8 of the patch should be 526.

Page 10 of the last issue of Software Notes contains a patch for CRT410.PRG. Please note that the line reading, "The hash before the patch should be 462-..." should read, "The hash before the patch should be 562-...."

SOFTWARE HINTS

Local Symbols In MACRO

It is possible to generate local symbols in MACRO definitions in the AM-100 MACRO assembler. The "\" operator, previously undocumented, allows you to implement local symbol generation for yourself. This operator evaluates the following expression, and appends the TEXT of the result to the preceding term. An example is presented below.

EXAMPLE:

```
   $=      0                                  ; local symbol counter

   ;Lower-case characters in string @R2 are converted to upper-case
   ;GTC: Get next character. Next character in string is retrieved to
   ;       see if it needs to be converted.
   ;STC: Store character. Character is placed back into string whether
   ;       converted or not. This increments the string pointer
   ;UCX: UCS exit. End of the MACRO. Restore saved registers
   DEFINE  UCS     STRING
           MOV     R1,-(SP)
           MOV     R2,-(SP)
   GTC\$:  MOVB    @R2,R1                     ; generates GTC0, GTC1, ...
           BEQ     UCX\$                      ; generates UCX0, UCX1, ...
           CMPB    R1,#'a
           BLO     STC\$                      ; generates STC0, STC1, ...
           CMPB    R1,#'z
           BHI     STC\$                      ; generates STC0, STC1, ...
           SUB     #40,R1
   STC\$:  MOVB    R1,(R2)+                   ; generates STC0, STC1, ...
           BR      GTC\$
   UCX\$   MOV     (SP)+,R2                   ; generates UCX0, UCX1, ...
           MOV     (SP)+,R1
   $=$+1                                      ; increment local symbol counter
                                              ; for next call to this MACRO
           ENDM
```

On the first call to the MACRO, the symbols GTC0, STC0, and UCX0 are defined. On the second call, GTC1, STC1, and UCX1 are defined. This is because the value of $ increments on each call.

Note that some symbol must be initialized as a local symbol counter. Before the MACRO is exited, this counter must be incremented by the largest value added in an expression to the counter, plus 1. For instance, all the symbols defined above could have been called by less mnemonic names, as follows: ..\$, ..\$+1, ..\$+2. In this case, the local symbol counter would have been incremented by three.

Also note that as these MACROs that generate local symbols are used, more and more symbols are generated. Remember that this only simulates local symbols. The symbols generated are real and take up space in the MACRO symbol table. When you are debugging a program with local symbols generated, the local symbols sometimes show up in the file at inappropriate times. It is possible for too many symbols to be generated, causing the assembler to run out of memory; or for DDT, and especially FIX, to be impeded by the symbol file.

At this time, there is a bug that prevents the result of the "\" operator from appearing in the list file. All that appears of the generated symbol is that portion to the left of the "\". The problem will be fixed in version 4.5.

Bank Switched Bitmaps

Section 2.8.1 of the document, "System Initialization Command File," in the 4.4 Release Notes states that the SYSMEM command can now be used to reduce the monitor size. It should be emphasized that if the bitmaps put in this switchable memory are not DSK devices, the driver for that device must be in system memory. The amount of monitor space being saved is the difference between the bitmap size and the driver size for non-DSK devices.

For example, if your system device is a Phoenix, with a double density floppy as a peripheral, part of the SYSTEM.INI might look like this:

```
SYSMEM 2:160000-177376
BITMAP DSK,1818,0,1,2,3,4,5/S
BITMAP DDA,154,0,1/S
SYSTEM TRM.DVR[1,6]
SYSTEM DDA.DVR[1,6]
SYSTEM
```

Notice, however, that the DDA.DVR has a size of 848 bytes and that the bitmap for the double density diskette is only 154 words (308 bytes). This increases the monitor size. In this case it is more advantageous to configure in the following way:

```
SYSMEM 2:160000-177376
BITMAP DSK,1818,0,1,2,3,4,5/S
BITMAP DDA,154,0,1
SYSTEM TRM.DVR[1,6]
;SYSTEM DDA.DVR[1,6]
SYSTEM
```

On the other hand, consider a Phoenix and Hawk system:

```
SYSMEM 2:160000-177376
BITMAP DSK,1818,0,1,2,3,4,5/S
BITMAP HWK,606,0,1/S
SYSTEM TRM.DVR[1,6]
SYSTEM HWK.DVR[1,6]
SYSTEM
```

The size of HWK.DVR is only 530 bytes, while the bitmap is 606 words (1212 bytes): a substantial saving in space. In fact, there is enough room left in the SYSMEM portion of memory to have separate bitmaps for each Hawk surface:

```
SYSMEM 2:160000-177376
BITMAP DSK,1818,0,1,2,3,4,5/S
BITMAP HWK,606,0/S
BITMAP HWK,606,1/S
SYSTEM TRM.DVR[1,6]
SYSTEM HWK.DVR[1,6]
SYSTEM
```

This doesn't save any more monitor space, but it does allow a faster disk access time without increasing the monitor size.

Of course, the driver for the DSK device need not be in memory, because it is imbedded in the monitor.

To Single Density Floppy Users

In view of last issue's patch to 200DVR.DVR, we should point out that any drivers implemented prior to this patch must have FIXDVR run to incorporate this patch in the driver that is used. Following is a list of hash totals for the drivers that have been generated by FIXDVR after the patch made to the 200 driver:

Persci STD	043-572-012-132
Persci AMS	725-555-432-035
Wangco STD	235-726-440-005
Wangco AMS	225-432-511-621

Of course, if any of these devices are system devices, you should run MONGEN again to include the new drivers.

Generating Monitors--Revisited

On page 5 in the last issue of Software Notes (June, 1980), we listed the hash totals of properly MONGENed system monitors. We wish to point out that these hashes will not be valid for version 4.4a. Because of the patch to the 200DVR.DVR, the four single density floppies must be re-MONGENed to include this driver. On the next page is a list of hash totals for SYSTEM.MON of the different system devices distributed by Alpha Micro. They include a patch to the SYSTEM.MON, reflecting "4.4a" instead of "4.4" when SYSTAT and SYSTEM commands are executed (see patch in SOFTWARE CHANGE NOTICES on page 12). The hash totals also include the new single density floppy drivers where applicable. There are no other changes to the monitor on the 4.4a SYSTEM.MON that are distributed by Alpha Micro.

Phoenix Monitor	411-771-243-462
Hawk Monitor	051-554-076-204
DDA Monitor	763-411-355-522
Persci AMS Monitor	144-342-744-117
Wangco AMS Monitor	412-047-010-366
Persci STD Monitor	244-324-512-205
Wangco STD Monitor	753-621-161-620

DSKCPY

It is important, when performing DSKCPY on Phoenix devices, to verify that the BADBLK.SYS on the destination surface is the correct file for that surface. Because the algorithm for DSKCPY on a Phoenix must take into consideration the alternate track information of the destination surface, there is not a literal copy from one surface to another. For example, if you are copying from SMD1: to SMD2:, DSKCPY first loads BADBLK.SYS of SMD2: into memory. Then it does a track by track duplication, referring to the BADBLK.SYS in memory to handle the possible alternate tracks of SMD2:. This means that it will copy the BADBLK.SYS of SMD1: onto SMD2:. Only upon completion of DSKCPY is the proper BADBLK.SYS SAVEd off. Therefore, if, during DSKCPY of a Phoenix, the system crashes, is rebooted, or is interrupted early enough by Control-C, SMD2: will have BADBLK.SYS from SMD1:. In this case, SMD2: must be recertified using the CRT410 program.

One way to tell if the proper BADBLK.SYS is present is to make note of the ID number assigned to the disk during the original CRT410 process. Traditionally, the serial number of the cartridge is used for removable packs, and the serial number of the drive followed by a dash and the platter number is used for fixed surfaces.

NOTE: DSKCPY on a Phoenix device can be executed only if both the source and destination surfaces have been certified with CRT410 under 4.4 or later versions of the software.

BADBLK.SYS and ERASE

While we're on the subject, we'd like to issue the warning that it is possible to inadvertently erase the file BADBLK.SYS. During wildcard COPYing, the system skips over BADBLK.SYS, with a warning that that file cannot be copied using a wildcard command. One might construe, then, that a wildcard ERASE will not erase BADBLK.SYS. However, this is not the case. A wildcard ERASE does include BADBLK.SYS in its deletion, and it gives no warning. The wildcard COPY command is the only program that issues such a warning. The loss of BADBLK.SYS means that all data on that surface is of doubtful integrity, and the surface must be recertified.

BASIC

DIMENSION statements in AlphaBASIC cannot be used to re-DIMension arrays. There has been some confusion on this fact because page 10 of the AlphaBASIC User's Manual (DWM-00100-01) states, "Once an array has been dimensioned by a DIM statement it may not be redimensioned by a subsequent DIM statement in the same program"; while page 35 of the same manual states, "Arrays may be redimensioned during execution of the program if desired." This apparent contradiction came about because there were two versions of BASIC: one that runs under the control of RUN.PRG, and interactive BASIC, which runs under BASIC.PRG. When the 3.3 monitor version was in use, you could re-DIMension arrays in the interactive BASIC only. That feature has since been taken out of interactive BASIC for the sake of compatibility.

Incidentally (as long as we're talking about arrays), though the AlphaBASIC User's Manual doesn't prohibit it, you cannot use MAP statements to set initial values of arrays. For example,

```
MAP Array (15),B,1,0
```

will not initialize the array and may cause other problems.

Miscellaneous

PROGRAM DESCRIPTION FOR TODCNV.PRG[1,4]

TODCNV is a routine you call to convert the time of day from ASCII to internal format or from internal format to ASCII. To do so, the TODCNV.PRG routine must be loaded into user or system memory (the code is reentrant).

Assuming that R5 contains the beginning address of the TODCNV program (obtained by either a FETCH or SRCH monitor call), here is how to call TODCNV to convert internal formatted time into ASCII text:

```
Parameters:
        R0              Flags (if set):
                           Bit <3>      No colon between hours and minutes.
                           Bit <4>      Use 12-hour time and append AM or PM.
                           Bit <5>      Omit seconds.

        R1              Destination pointer:
                           Pointer to buffer area.  If R1 is zero, the string
                           is output to the user's terminal.

        R2, R3          Time to be output:
                           Time stored as number of clock ticks since
                           midnight.  If R2 and R3 are zero, the current
                           system time is output.

Returned:
        R0              Preserved.
        R1              Destroyed
        R2              Updated destination pointer.
        R3, R4, R5      Destroyed.

Method of call:
        CALL    @R5
```

Under the same assumption, here is how to call TODCNV to convert ASCII text into internal formatted time:

Parameters:

R0	Flags (if set):		
		Bits <0-1>	Inclusion of seconds in field:
			Both off -- optional
			<0> only -- illegal
			<1> only -- mandatory
		Bits <2-3>	Inclusion of AM/PM in field:
			Both off -- optional
			<2> only -- illegal
			<3> only -- mandatory
R2	Source pointer		

Returned:

R0	Error codes:		
		0	Successful conversion
		1	Illegal time format.
R1	Destroyed.		
R2	Updated destination pointer.		
R3, R4	Time in internal format (ticks since midnight).		
R5	Destroyed.		

Method of call:

```
        CALL    2(R5)
```

Special notes of interest:

Parse is completed by AM/PM, carriage return, or null. Spaces and tabs are ignored. If an error condition occurs, the destination pointer is left pointing to the character immediately following the one that caused the error.

SOFTWARE CHANGE NOTICES

SYSTEM.MON (To specify "4.4a" in Monitor)

The following patch to SYSTEM.MON places an "a" into the monitor to reflect Version "4.4a" instead of "4.4," when SYSTAT or SYSTEM programs are executed. For the sake of hash total checking, we started with a re-MONGENed SYSTEM.MON, then incorporated the patch. This new SYSTEM.MON may be used itself or as a core for another MONGENed monitor. This patch is included in the 4.4a system disk that may be ordered from Alpha Micro.

```
.LOG SYS: RET
.DDT SYSTEM.MON RET
PROGRAM BASE IS xxxxxx
PROGRAM SIZE IS 32722

12/     LCC 4  60464 RET
^C
.DIR MEM:/H RET
SYSTEM          MON 13778   xxx-xxx-xxx-xxx              MEM:
.SAVE SYSTEM.MON RET
ERASE SYSTEM.MON, SAVE SYSTEM.MON
.DEL * RET
SYSTEM.MON
```

The correct hash total you should see is listed in the section, "Generating Monitors--Revisited," on page 8.

SYSTEM.MON (To correct file service problem)

The following patch corrects a possible problem with the file service routine. It is not considered a critical bug since most programs check for this condition before the file service routine is called. This patch is not included in AMOS version 4.4a.

```
.LOG SYS: RET
.DDT SYSTEM.MON RET
PROGRAM BASE IS xxxxxx
PROGRAM SIZE IS 32722

11046/  TST 4(R2)       TST 4(R4) RET
^C
.DIR MEM:/H RET
SYSTEM MON 13778 xxx-xxx-xxx-xxx                MEM:
.SAVE SYSTEM.MON RET
ERASE SYSTEM.MON, SAVE SYSTEM.MON
.DEL * RET
SYSTEM.MON
```

The following list shows how the patch remedying the file service problem will change the 4.4a monitors:

Monitor Type	4.4a Hash Total	Patched 4.4a
Phoenix Monitor	411-771-243-462	035-464-022-200
Hawk Monitor	051-554-076-204	164-340-510-147
DDA Monitor	763-411-355-522	424-223-630-315
Persci AMS Monitor	144-342-744-117	775-457-012-350
Wangco AMS Monitor	412-047-010-366	763-405-242-214
Persci STD Monitor	244-324-512-205	164-370-132-304
Wangco STD Monitor	753-621-161-620	335-564-116-104

Please use these hash.totals for comparison with your own after implementing the patch. You will get the hash total listed under "Patched 4.4a" above, if you actually patch the monitor or if you use a patched monitor as the core of a MONGENed monitor.