JULY '86

The San Leandro Computer Club for Atari Microcomputers is an independent, non-profit organization and users' group with no connection to Atari Incorporated. Membership fees are $20 per year. Membership includes access to the computer library, subscription to the Journal, and classes when held. Permission to reprint articles in any non-commercial publication is permitted without written authorization, provided proper credit is given to the San Leandro Computer Club and the author. Opinions expressed are those of the author and do not necessarily represent the views of SLCC.

EDITOR	Jerry Telfer	233-9158
ASSISTANT EDITOR	Mark Blum	483-8166
ART DIRECTOR	Jim Hood	534-2197
CALENDAR INFORMATION	Tom Tisby	352-4482
JOURNAL COPYING	Mark Blum	483-8166
JOURNAL ADVERTISER	Ron Seymour	537-3183

JOURNAL ADVERTISING RATES

FULL PAGE:	$40.00	QUARTER PAGE:	$10.00
HALF PAGE:	$20.00	BUSINESS CARD:	$ 5.00

PRESIDENT	Bob Barton	352-8118
VICE-PRESIDENT	Mike Sawley	482-5061
TREASURER	Lois Hansen	482-2222
SECRETARY	Jim Moran	523-4231

TAPE LIBRARIAN	Frank Hand	638-6032
ASSY LANGUAGE SIG	Frank Daniel	471-8133
BASIC LANGUAGE SIG	Guy Cochrane	582-5561
BEGINNERS SIG	Richard Stiehl	835-9857
MISCELLANEOUS SIG	Phil Mitchell	351-2208
ATR8000 SIG	Mike Sawley	482-5061
LOGO SIG	Lois Hansen	482-2222
ACTION! SIG	Jim Warren	782-7090
ST SIG	Bob Barton	352-8118
C SIG	Dave Beckemeyer	658-5318
BBS SIG	Mike Sawley	482-5061
PROGRAM CHAIRMAN	Richard Scott	887-8357
PRINT LIBRARIAN	Einar Andrade	484-4484
DISK LIBRARIAN	James Moran	523-4231
ST SOFTWARE CHAIRMAN:		
	Mike Curry	782-9148
8 BIT SOFTWARE CHAIRMEN:		
	Mark Perez	792-0398
	Cliff Schenkhuizen	537-5245

KEY SYSTEM BBS: (415) 352-5528
Official 8-BIT BBS of the SLCC
SYSOP: Mike Sawley
THE VILLAGE BBS: (415) 783-5545
Official 16-BIT BBS of the SLCC
SYSOP: Mike Curry

IN THIS ISSUE: no longer cowed by the RAM-happy ST owners, 8-bit Atarians are beginning to see a number of impressive memory upgrades. There are some things to watch out for, but a half-meg 8-bit is feasible. SLCC and the SLCC JOURNAL hold ourselves blameless if you fry your computer doing any of the nifty upgrades in this issue.

Also, we have the debut of our "spy in the sky" column on the plethora of ST rumors. There is a reader survey that we hope you will get to us so we can meet your needs in the JOURNAL. Our thanks to Chris Stewart for the source photo used in producing this month's cover. Enjoy!

CONTENTS

OOPS!

Last month's article on "The 80 Column Atari" by Mike Bellante, credited to The Key System, came to SLCC from Diablo ACE (DACE). We regret the omission of credit to the fine folks at DACE.

CORRESPONDENCE ADDRESS & NEWSLETTER EXCHANGE
SAN LEANDRO COMPUTER CLUB
P. O. BOX 1525
SAN LEANDRO, CA 94577-0152

EDITORIAL

The user group movement is a powerful force fed mainly by one source -- the sharing of information.

There are many means used to achieve this goal in the SLCC. Monthly main meetings, SIGs, the Key System and Village BBSs, numerous phone calls between members with common interests/problems, and the SLCC JOURNAL.

During the months ahead it will be our goal to:
maintain the standard of excellence set by Ron Seymour and Tom Bennett during their years of producing this JOURNAL,
achieve timely delivery of the JOURNAL to the membership,
redesign the JOURNAL to modern magazine appearance,
maintain an overall balance in our coverage of all Atari systems,
reprint the very best of those articles published by other club newsletters,
and most importantly, to improve the ratio of SLCC-produced material published in the JOURNAL.

This last item in the list of goals is *crucial* to the continued improvement of SLCC's standing among Atari user groups. SLCC now has over 400 active members, making it one of the largest Atari user groups in the world. *Our ST SIG alone is larger than many clubs' total membership.* This offers a potential wellspring of new Atari knowledge. We wish to tap that wellspring at all levels.

There is a temptation among new members to sit quietly at the monthly meeting, completely certain that they are the only persons in the room who are not rip-roaring heavy duty hackers able to create full-blown major programs (bug-free, of course) between dessert and the Cosby show. This reticence is well understood by us, since we have shared it in our time. It has, however, no basis in fact.

The freshest novice may not have learned the "correct" way to achieve a programming goal, and thus may in innocence contrive a solution that matches or outperforms the textbook standard. This information should be shared.

There is not one member of the SLCC that doesn't have something of value to contribute to the JOURNAL, and thereby to the club. We will publish listings of original programs, fragments of code which do something interesting or amusing, reviews of software or hardware, explorations of undocumented/poorly documented program capabilities, graphics screens, questions about Atari computing and hopefully their answers, and just about anything else that interests the members. We will accept member contributions via the club's BBSs, on diskette, or by phone if necessary; *but we want your contributions.* We will also assume the task of helping to polish member submissions into publishable articles. You don't have to be a hacker Hemingway, you only need some ideas to share.

We do not wish to edit a "Reader's Digest of Atari Computing" - we want to produce the SLCC JOURNAL, and with your help make it the premier source among Atari user group publications.

Jerry Telfer
(130XE)

Jim Hood
(800 & 520ST)

Mark Blum
(1040ST)

DANGERS OF NON-STANDARD MEMORY EXPANSIONS

by Bill Wilkinson, OSS

This technical note is being written because so many of the memory expansion schemes I see being touted are NOT compatible with a standard 130XE. If you implement the memory expansion per most of these schemes, you will be missing one important feature of the 130XE: the ability to direct ANTIC to do its DMA to either main memory or the requested bank of memory.

In a standard 130XE, clearing bit 5 to zero requests ANTIC to follow the bank switching; setting bit 5 to a one tells it to remain in main memory, no matter what memory bank was requested.
This is an important feature! Mark Rose (also of OSS) and I will take credit for being instrumental in the creation of this bit.

When Atari asked us to do DOS 2.5 and its RamDisk, their prototype hardware had ANTIC following bit 4 along with the CPU. The most obvious problem with this is that you can't use the extra banks for CPU purposes (e.g., RamDisk) when ANTIC is doing its DMA in the memory between $4000 and $7FFF. The problem was especially acute with AtariWriter (the 16K cartridge version), since its display memory is ALWAYS in this range.

Actually, Mark and I found that if you are ONLY using the bank select memory for a RamDisk, this is not an onerous restriction. It simply means that you could only do pseudo-sector transfers during vertical blank. And, in fact, DOS 2.5 still has a flag in it which you can POKE which will tell it to only use extended memory during deferred vertical blank.

Now, there was another hardware solution, which we mentioned to Atari: simply never allow ANTIC to use extended memory. We discussed the two options with Atari, and both they and we decided we felt strongly that the capability of bank selecting ANTIC's memory was important. Thus the use of that bit.

So, if your 800XL hardware mod works with the AtariWriter cartridge, then you obviously adopted that second hardware solution: don't let ANTIC use extended memory. That is not a really terrible decision (especially if it is economically motivated), but it does mean that it is possible that some future 130XE software will not run on your modified machine. (Actually, I already have at least one piece of software, written in ACTION!, which depends on the 130XE's method. But it's only an ultra-fast picture switching demo, so it's no big deal.)

There is a mod to both the 130XE and 800XL which maintains the 130XE/ANTIC bank select capability. It was designed by Charles Andrews of Eugene, Oregon, and he showed a 320XE using this mod at CES in January (in Atari's booth, as a courtesy to him by Atari--though it does appear to be an implicit endorsement of his scheme). I believe his method uses an entirely separate port for controlling the beyond-130XE extensions (in the $D6xx range, maybe?). However, I devised a method of doing the same thing using only PortB. The scheme is outlined in the following paragraph.

A "LEGAL 320XE:

This mod depends on the fact that the diagnostic ROM area is only used at powerup or by the self-test routines. At these times, both ANTIC and the CPU are using only main memory, so bits 4 and 5 of $D301 are both set to one. Thus we change the "enable" of the diagnostics from the logic equation

```
diag_enable = not_bit7
```

to

```
diag_enable = not_bit7 and bit4 and bit5.
```

Then the enable for the extended RAM becomes RAM_enable = (not_bit4 or not_bit5) and we can now use bits 6 and 7 for bank selection in the same manner that other schemes use bits 6 and 5. Reason this works: even if Atari ever changes the self-diagnostics so that they check the extended RAM, they can't put that particular code in the ROM which overlays $5000-$57FF, because that's right in the middle of the RAM area they need to check!

576K Mod
Iuscle for your 130XE

by SCOTT PETERSON

Here we go again, this time I recommend you have some electronics experience if you wish to perform the upgrade. Some of the work is duplicated from the 320K upgrade so 320XE owners will not have as much work to do. One other point, when in the 576K mode you MUST use some sort of BASIC cart. as you lose the internal BASIC, this is only in the 576K mode, in the 130XE mode internal BASIC will function normally.

TOOLS NEEDED; To perform this upgrade you need the following; Low wattage fine tip soldering iron. Vacuum de-soldering tool (like Radio Shack PN#64-2098). Some 30-gauge wire (Radio Shack PN#278-501). #2 Phillips head screwdriver. Heat-shrink tubing, 1/8 in. diameter Also a pair of small needle-nose pliers and a small flat tip screwdriver are handy.

PARTS NEEDED:

Z1	74LS158
Z2-Z17	41256 (150ns.)
Z18	74LS138
Z19	7432
R1-R2	33 Ohm 1/4 watt resistor.
S1	Micro-mini DPDT switch (like Radio Shack PN#275-626)

Remove the 130XE case and metal RF shield to get down to the mother board. (320XE users go to step two).

STEP ONE: Now de-solder and remove the eight RAM chips U26 thru U33 (MT4264). They are the row closest to the TV RF module (do NOT use solder wick, the circuit board of the 130XE has very weak runs and they will pull loose if not completely de-soldered). Replace these with the 16 pin low profile sockets. Take a piece of wire approximately 12 in. long and run a jumper from pin 1 of each socket to the next. When you are done, the wire should be attached to pin 1 of each of the new sockets and you should have about 6 inchs left over. Do this on the rear of the motherboard and then snake the wire thru the large hole near the RAM chips.

Next, desolder and remove U23(CO14795), and replace it with a 40 pin socket. Bend up pins 15 and 16 on U23 and insert it in the socket you just installed. Take Z1(74LS158) and break off pins 5,6,7,9,10,11,12,13,14. Bend up the other pins on it except 8 and 16. Put this "piggy back" on top o U20(HD14050) and solder pins 8 and 16 of Z1 to pins 8 and 16 on U20. Now take a short jumper from pin 15 on Z1 to pin 8 of Z1 Take a piece of wire about 4 in. long solder one end to pin 30 on the chip marked "CO14805" on the motherboard, and the other end to pin 1 on Z1.

Next solder a wire to pin 15 (one of the two you bent out) of U23 and connect the other end to pin 2 on Z1. Solder a wire to pin 16 on U23 and connect the other end to pin 3 on Z1. Take R1(33 Ohm) and trim the leads to about 1/4 in. Take the wire you connected to pin 1 of the RAM chip sockets and solder it to one end of R1, solder the other end of R1 to pin 4 on Z1.

STEP TWO: Slide the motherboard back into the bottom half of the plastic case (do not use the RF shield, you must be able to get at the motherboard), and attach the keyboard. It will rest above the motherboard without touching it. Test all 41256 RAM chips by putting one set of 8 in the sockets and using the handlers (or DOS's), and then the next. After testing all RAM chips remove them all from the sockets, and take 8 of them and cut about half of pin 15 off of each one. Only the "fat" part of pin 15 should be left. After doing this you have to "piggy back" the 8 256K RAM chips with the short pin 15's on top of the other 8 256K RAM chips. Now solder all the pins together on the stacked RAM chips except for pin 15. It should not be touching the other pin 15. Make sure you have them going pin 1 to 1, pin 2 to 2,etc. When you get done you will have 8 sets of Piggy backed 256K RAM chips. Now take a piece of wire about 16 in. long and run a jumper from pin 15 to the next one on all the top 256k DRAM's, leaving about 1 inch between each RAM chip.

Put these stacked RAM chips into the 8 sockets you installed earlier. Take Z18(74LS138) and bend up all the pins except 8 and 16, cut the pins you bent up in half so only the fat part is left, and solder pins 8 and 16 to pins 8 and 16 of the other 74LS138 right below the U23(CO14795). Take Z19 and bend up all pins except 7 and 14, once again cut all the pins you bent up in half and solder pins 7 and 14 to pins 7 and 14 of the 74LS08 right below U23. Take the wire you jumpered earlier to pin 15 of Z10 thru Z17(the upper row of 256K RAM chips) and go out 2 in. and cut the wire, n install R2(33 Ohm) between this cut. Place piece of heat shrink tubing over R1 and make sure no wire is exposed and heat it with a lighter. Take the other end of this wire and connect it to Z18 pin 14. Find the two 33 Ohm resistors just to the right of U28 (one of the RAM chips you socketed). The upper one of the two is R111, desolder the right leg of it and bend it up. Take a piece of wire and solder it to the land where you just removed the leg of R111. Connect the other end to Z18 pin 4. Trim back the leg of R111 and solder a wire to it, slip a piece of heat shrink tube over it and heat it up.

Now connect the other end to Z18 pin 12. Take a short wire and run a jumper from pins 1 and 16 of Z18. Take another short wire and connect a jumper from pins 3, 5, and 8 of Z18. Now connect a wire from Z18 pin 2 to Z19 pin 3. Find the wire you installed from U23 pin 15 to Z1 (74LS158) pin 2 and desolder it from U23. Take it and reconnect it to Z19 pin 11. Ok, now pry U23 (C014795) back out of the socket and bend up pin 11, plug it back in. Run a jumper from pins 1 and 4 of Z19, and another jumper from pins 10 and 13 of Z19. Connect a wire from U23 pin 11 to Z19 pin 1, and from U23 pin 15 to Z19 pin 13. Now connect a wire from Z19 pin 8 to the right side of the 3.3K Ohm resistor marked R206 (located at the bottom right of U23). Connect a wire to Z19 pin 6 and run it to pin 18 of U3 (C061618). Now comes the tricky part, drill a small hole (1/4 in. or so, depending on the switch size) at the rear right on the back of your 130XE. Take the small DPDT switch (S1) and install it in the hole. Now connect it as shown (make sure the switch DOESN'T have a center off position);

```
                 S1(rear)
   U23          _________           U23
 pin 20 ----:-O      O-:------ pin 1
            :  \    /  :
   Z19   ---:-O \/ O-:---      Z19
pins 2+12   :    /\    :     pins 5+9
            :  /    \  :
            : O      O :
            :__________:
```

Note: where the wires cross in the middle, they are NOT connected. Make the connection from the switch to U23 on the rear of the motherboard. Well that's it(thank God). Now re-assemble the computer, being careful not to break any wiring going to the switch. You should now have in one switch position a 100% compatible 130XE, and in the other you have a 576K 130XE that does not have Antic memory enhance mode and also cannot use internal BASIC. In the 130XE mode you gain 64K as bit 6 of the PIA can still be used. The following page list of the bit table and numbers to be used in location 54017 (PORTB). Once again, if you need, help call the Peanut Gallery BBS (408)-384-3906. If you want a mailer of all the upgrades I have, as well as a disk with handlers, source codes, etc. send a money order (please, no checks) for $10.00 to; Scott Peterson P.O. Box 33 Ft.Ord, CA. 93941-0033. This includes the 800 288K upgrade by D.G.Byrd, the 800XL/256K (C.Burchholz), the 130XE/320k upgrade and anything else I finish. Good luck, and have fun. Memory Control Register 54017 (D301)

130XE in 576K mode.

```
Bit 7 6 5 4 3 2 1 0
      D a b C c d e R
```

D=0 enable diag. ROM
R=1 enable OS ROM
C=0 enable extended memory
abcde= memory control bits.

Bank#	Control#(dec)	Hex
Bank 0	---------->129	81
Bank 1	---------->131	83
Bank 2	---------->133	85
Bank 3	---------->135	87
Bank 4	---------->137	89
Bank 5	---------->139	8B
Bank 6	---------->141	8D
Bank 7	---------->143	8F
Bank 8	---------->161	A1
Bank 9	---------->163	A3
Bank 10	--------->165	A5
Bank 11	--------->167	A7
Bank 12	--------->169	A9
Bank 13	--------->171	AB
Bank 14	--------->173	AD
Bank 15	--------->175	AF
Bank 16	--------->193	C1
Bank 17	--------->195	C3
Bank 18	--------->197	C5
Bank 19	--------->199	C7
Bank 20	--------->201	C9
Bank 21	--------->203	CB
Bank 22	--------->205	CD
Bank 23	--------->207	CF
Bank 24	--------->225	E1
Bank 25	--------->227	E3
Bank 26	--------->229	E5
Bank 27	--------->231	E7
Bank 28	--------->233	E9
Bank 29	--------->235	EB
Bank 30	--------->237	ED
Bank 31	--------->239	EF

Testing Extended Memory

by Larry Copenhaver

With the large number of 130XE computers out there, and the increasing number of expanded 800XL and 130XE computers, here's a "down-and-dirty" (written in BASIC) memory test.

It will only check the EXTENDED RAM (the normal RAM can be checked with the built-in diagnostics).

The method is very slow and simple. We just set all the bits to off (POKE it with zero). Then check to see that they are in fact all off. We then turn on all bits (POKE it with 255) and check to see if they are all on. If any discrepancy is found we report that error and end the program.
Of course the extra RAM in these machines is accessed by windows of 16K each and we must make visible to the computer these windows (or banks) as they are needed by POKEing the appropriate value in PORTB(54017).

This little program was written on the night before the printing of this newsletter (KC-ACE) on the "prototype" 320K 130XE. And used to check all 16 banks of 16K each on this computer (it took about 3 hours to run).

This was really just an interim program to help anyone that thinks they might have a RAM problem. Watch for next month's newsletter. I will have an assembly language version that will be much faster.

(reprinted from Feb '86 KC-ACE newletter, Kansas)

```
10 REM EXTENDED RAM TEST
20 REM ****************
30 REM BY LARRY COPENHAVER
40 REM ******************
50 GOSUB 1000
54 POKE 82,3:GRAPHICS 0
60 PRINT "WARNING--THIS WILL TAKE A LONG
 TIME"
65 PRINT "(IT IS WRITTEN IN BASIC, YOU K
NOW)"
70 PRINT :PRINT :FOR WAIT=1 TO 900:NEXT
WAIT
71 PORTB=54017:BANKSTART=16384:BANKEND=3
2767:PBN=PEEK(PORTB)
74 REM *****************
75 REM SEE THAT 4 BANKS ARE THERE
76 REM *****************
77 RESTORE LINE
80 FOR X=1 TO BANKCNT
90 READ BANKVALUE
100 POKE PORTB,BANKVALUE
110 POKE BANKSTART,X:NEXT X
130 RESTORE LINE
140 FOR X=1 TO BNKCNT
150 READ BANKVALUE
160 POKE PORTB,BANKVALUE
170 CHECK=PEEK(BANKSTART)
180 IF CHECK=X THEN PRINT "BANK ";X;" IS
 THERE!!":GOTO 200
190 PRINT "ERROR IN BANK ";X:POKE PORTB,
PBN:END
200 NEXT X
210 POKE PORTB,PBN
220 REM ***************
230 REM NOW TURN OFF ALL BITS
240 REM ***************
250 RESTORE LINE
260 FOR X=1 TO BNKCNT
270 PRINT "TURNING OFF ALL BITS IN BANK
";X
280 READ BANKVALUE
289 REM FILL IT WITH ZEROS
290 POKE PORTB,BANKVALUE
300 REM FILL IT WITH ZEROS
310 FOR Y=BANKSTART TO BANKEND
320 POKE Y,0
330 NEXT Y:NEXT X
350 REM **************
360 REM NOW CHECK ALL BITS
370 REM **************
380 RESTORE LINE
390 FOR X=1 TO BNKCNT
400 PRINT "CHECKING ALL BITS IN BANK ";X
410 READ BANKVALUE
420 POKE PORTB,BANKVALUE
430 REM CHECK FOR ZEROS
440 FOR Y=BANKSTART TO BANKEND
450 A=PEEK(Y):IF A=0 THEN GOTO 480
470 PRINT "ERROR IN BANK ";X:POKE PORTB,
PBN:END
480 NEXT Y
490 PRINT "NO ERRORS SO FAR IN BANK ";X
500 NEXT X
510 POKE PORTB,PBN
520 REM *************
530 REM NOW TURN ON ALL BITS
540 REM *************
550 RESTORE LINE
560 FOR X=1 TO BNKCNT
570 PRINT "TURNING ON ALL BITS IN BANK "
;X
580 READ BANKVALUE
590 REM FILL IT WITH ONES
600 POKE PORTB,BANKVALUE
610 FOR Y=BANKSTART TO BANKEND
620 POKE Y,255
630 NEXT Y:NEXT X
650 REM *************
660 REM NOW CHECK ALL BITS
670 REM ************
680 RESTORE LINE
690 FOR X=1 TO BNKCNT
700 PRINT "CHECKING ALL BITS IN BANK" ;X
710 READ VANKVALUE
720 POKE PORTB,BANKVALUE
730 REM CHECK FOR ONES
740 FOR Y=BANKSTART TO BANKEND
750 A=PEEK(Y)
760 IF A=255 THEN GOTO 480
770 PRINT "ERROR IN BANK ";X:POKE PORTB,
PBN:END
780 NEXT Y
790 PRINT "NO ERRORS SO FAR IN BANK ";X:
NEXT X
810 POKE PORTB,PBN
820 PRINT :PRINT "ALL PASSED!!!":END
900 DATA 129,133,137,141
910 DATA 129,133,137,141,161,165,169,173
,193,197,201,205,225,229,233,237
920 DATA 129,133,137,141,193,197,201,205
,225,229,233,237
930 DATA 161,165,169,173,193,197,201,205
,225,229,233,237
1000 GRAPHICS 0
1010 ? :? :? "WHICH COMPUTER IS THIS?"
1020 POKE 82,6:?
1030 ? "1. 130XE(STANDARD)"
1040 ? "2. 130XE(320K)"
1050 ? "3. 800XL(NEWELL VERSION)"
1060 ? "4. 800XL(KCACE VERSION)"
1070 TRAP 1000:INPUT ANSWER
1080 ON ANSWER GOTO 1100,1200,1300,1
1090 GOTO 1000
1100 LINE=900:BNKCNT=4:RETURN
1200 LINE=910:BNKCNT=16:RETURN
1300 LINE=920:BNKCNT=12:RETURN
1400 LINE=930:BNKCNT=12:RETURN
```

An 800 OS for the 1200XL

Source: RAM, September 1985
Reprinted from ATARI SIG, MDD RCC

1200 XL to 800 OS

Bummed out with the hassles built into the 1200XL OS? Want burn your translator disks, but can't shell out the $80 for BOSS XL? Follow these simple instructions to replace the nasty 1200XL OS with the nice old 400/800 OS for $16.50 or less!

1. Pull the Rev. B Roms out of the motherboard of a 400 or from the 10K ROM of an 800. Make sure the chips function properly.

2. If you can't find a set of ROMs, order them from Amercian TV 15338 Inverness St., San Leandro, CA 94579, 415-352-3787.

3. At this point you should have three chips, marked C012399B, C012499B, and C014599B.

4. Place the C014599B piggyback on the C012499B with the notched ends facing the same direction. Solder pin 1 to pin 1, pin 2 to pin 2, etc., for all 24 pins.

5. Open the 1200 XL using a small Phillips-head driver on the screws and a needlenose pliers to pull the pop rivets holding the RF shield together. This step is completely nondestructive. There is nothing to cut, unsolder, or mutilate.

6. Locate the two 24-pin ROM chips on the 1200 XL board near the cartridge slot. Notice they are marked U12 and U13 in white letters on the board. Also pay attention to which way the notched ends of the chips are facing.

7. Remove these ROM chips and insert the chip marked C012399B into U12 and the piggybacked chips into U13.

8. Test the board before putting the 1200 back together.

You now have an 800 with a 1200 keyboard. This modification was created by Brent Borghese of A.C.E. of Columbus Ohio.

Mike Sawley

ACTION! LANGUAGE REVIEW

The Action! Language by Optimized Systems Software is a non-portable language that generates very fast object code for the Atari 800 and 800XL home computers. It is the fastest high level language available for these machines in both compilation and execution speed. It runs 100 times faster than Atari BASIC and executes more than twice as fast as Kyan Pascal.

Compilation time in Action! compared to Kyan Pascal is many times faster for several reasons. The Pascal compiler is large and requires a relatively long load time wherein Action! is already present on a ROM cartridge. Pascal makes type checks on 99% of the variable names and type declarations while Action! guarantees type compatibility among its standard predefined types; no further type checking is done. Kyan Pascal uses a two pass compiler and Action! does it in one. The design of the Action! system allows both compilation from memory to memory, memory to disk, and disk to memory while Kyan Pascal compiles from disk to disk only because of the large size of the compiler. Action! even allows two different source codes (programs or subroutines) to reside in memory and can compile and run either from the monitor without building entire programs around them.

...continued on next page

Many of the features of the prominent high level languages are found in Action! such as block structure, parameter passing, IF-THEN-ELSE nesting, record declaration, and all of the high level loop constructs (no GOTO available!). Generally, if the feature contributed to the concept of structured programming while not slowing the execution of the object code (could be implemented with a minimum of output code) - the feature was put in. Other features must be defined if they are required. It has been said that the construction of the Action! Language was explicitly designed to optimize 6502 instruction set and that is the reason for its fast execution speed. And the object code generated by Action! reads like that of a 6502 assembly language programmer (without the comments, of course!).

One thing that I found very discouraging was that recursion does not return popped values. Parameters are overwritten if called by the same subroutine. This is because a block of memory is used explicitly for variables and the names of the callers. The one pass compilation is another factor in this limitation. Recursion can be simulated, so I am told, and I am looking for a demonstration of this useful technique. Action! uses only the 256 byte hardware stack provided for the 6502 microprocessor to save return addresses from procedure and function calls during the run. While this is a factor in reducing the speed of execution, some of the utility of the language has been compromised.

The standard types supported are: BYTE (8-bit), CARD (16-bit used as addresses or large positive values), INT (-32768 - 32767), CHAR (256 characters in the set), and POINTER (for passing by reference etc.). Any coercions desired can be made between any of these standard types even though the lengths vary between 8 and 16 bits. FLOAT is an option which is not included in the basic language but can be added at extra cost. This means that division truncates the decimal part. ARRAY is a standard structure and always numbers from 0 to n-1. It can contain any of the standard types. A special TYPE declaration can be made to declare records of standard types. But arrays of records must be implemented using a template which is moved over a larger array calculated in advance by multiplying the record size (in bytes) times the number of records. In this case, aliasing is a necessary evil. Dynamic array length declaration may not be done, but array bounds are not checked during execution. In fact, array length need not be specified all. However, the user must insure th data does not collide with the program .. this option is used.

Assignment in the type declaration is allowed. However, if the value is not enclosed in square brackets, the assignment is assumed to be a binding of an address to the variable name itself. Also, arrays may be bound to any feasible starting address by the programmer by the same technique. Since overflow is not checked, any values assignment during compilation or execution will be made modulo 256 or 65536 depending on the type of the variable being assigned to.

Overloading of operators is generally not done, but Action! has some other strange peculiarities. There are some operations which have more than one symbol to mean the same thing! "Not equal" can be represented by either <> or #. "and" can be either AND or &. "Or" either OR or %. "Exclusive-or" either XOR or !. And BYTE is a CHAR is a BYTE? All of this interchangeability is logical but strange when the size of the language was originally limited because of speed considerations.

Procedures and functions that return values are an important part of the language. Their form is somewhat similar to Pascal except that nesting of them is not allowed. Any subroutine calling another must call above due to the one pass restriction. The reserved word RETURN is used at the bottom of procedures and functions to indicate their termination. A missing RETURN is not a compiler error. The BEGIN, END, and semicolon are not used as delimiters. Instead, an EOLN separates each line logically. The semicolon is used as in assembly language to indicate that a comment follows.

Eight parameters maximum may be passed to a subroutine. This is a violation of the "Zero-One-Infinity Rule," but microcomputers do have limitations. Also, there is a restriction to not use channel 7 for input-output as that is reserved for the keyboard only by the Action! system.

Over all, the language is designed for the use of experienced programmers who need the the flexibility to manipulate t' operating system and the ability to wr fast, readable code. Then, the full capability of real-time graphics on the Atari 8-bit machine can be realized.

8-bit

Disk of the Month

16-bit

First off, before you start sending us packages that tick, late night phone calls, and making voodoo dolls in our likeness, we would like to take this opportunity to correct a misnomer on the last disk-of-the-month. "ERRORS?....the DYNAMIC DUO?!?" Well, not entirely.

Contrary to popular belief, Disk Label Construction Set will run under normal Atari Beginners All-purpose Symbolic Instruction Code.... (BASIC to you and me.) What does this all mean to you? This:

```
1 DIM S1$(60),S2$(60),C1$(60),C2$(60),T$(60
),N$(5)
20 GOSUB 370
276 GRAPHICS 0:POKE 710,0
```

Type in the above lines after the program has loaded and save it back to disk. It will now run with normal BASIC.

Now that we foiled that sorry excuse for a Communist take-over, and Mark Blum has just figured out what we have been talking about, let's get down to business.

Spanning the globe to bring you the latest in public domain software, the Dynamic Duo presents the July 1986 disk-of-the-month. As usual this disk is nothing short of spectacular. A veritable smorgasbord of quality software. But alas, for security reasons, we can only mention a few of these programs-

BINARY FILE TITLER - This program was written by the infamous Keith Ledbetter, author of "850 Express!," and it will display a standard ATASCII text file before running a binary file.
PRINTSHOP REFERENCE CARD MAKER - This program will take a standard PRINTSHOP picture disk and print out the pictures in an easy-to-read reference card format.
EASY-CARD DATABASE - An original by the Dynamic Duo. This database is a simulation of the popular ROLODEX file system.

.....and many others, including a memory checker for extended-RAM 130XEs and some graphics and sound demos...
Note: the above characters described in the above article are purely fictional. Any similarities to any person, alive or dead, past, present, or future, is purely coincidental. This newsletter will self destruct in five seconds.

June 86 DOM #1

---------Zoomracks Demo Disk ------(8606A)

GETST1.ZRX	GETST2.ZRX	GETST3.ZRX
MACAUTO.ZRX	QAPPLICT.ZRX	QAPT.ZRX
QCHECK.ZRX	QDEMO.ZRX	QINVNTRY,ZRX
QLETTERS.ZRX	QNAMES.ZRX	QNOTES.ZRX
QPLETTER.ZRX	QPROJEC.ZRX	QRECIPE.ZRX
QSALES.ZRX	QSCHOOL.ZRX	QTICKLER.ZRX
ZR.DAT	ZR.HLP	ZRDEMO.IMG
ZRDEMO.PRG!	ZRDEMO12.386%	

June 86 DOM #2

--- Micro Emacs 3.6 Source Code --(8606B)

IKBD1.DOC	IKBD2.DOC	LIST.DOC
LIST.TTP!	MEMTEST.INF	MODEM7.DOC
MODEM7.TOS!	MOUSE.PRG!	PICSWITC.PRG
PSWITCH.DOC	SPOOL33K.PRG	
A:\STMAKE:		
CTYPE.H	MAKE.C	MAKE.DOC
MAKE.PRG!	MAKFILE	MAKFILE2
MSTRING.C	MSTRING.H	
A:\UEMACS.36:		
BASIC.C	BUFFER.C	C.BAT!
CINFO.C	DEF.H	DISPLAY.C
ECHO.C	EXTEND.C	FILE.C
FILEIO.C	GETN.S	KBD.C
LINE.C	MAIN.C	RANDOM.C
REGION.C	SEARCH.C	SPAWN.C
SYMBOL.C	SYSDEF.H	TTY.C
TTYDEF.H	TTYIO.C	TTYKBD.C
VERSION.C	WINDOW.C	WORD.C
XEMACS.BAT!	XEMACS.INP	

June 86 DOM #3

----------- PROFF Disk -----------(8606C)

CARTPORT.INF	HELPER.PRG!	SPOOL.ACC
A:\PROFF.SRC:		
DEBUG.H	DECL.H	DEFS.H
DOSTUFF2.C	EMACS.TTP!	EVAL.C
LOOK.C	LOOKUP.C	LOOKUP.H
LTB.C	MAKE.SH!	MAP.C
PINIT.C	PROFF.C	PROFF.H
PROFF.PRG!	PROFF01.C	PROFF02.C
PROFFMAN.OUT	PROFFMAN.PRF	PROFFSYM.NEW
PUTWRD.C	PXLEX.C	PXXPARSE.C
STACK.C		

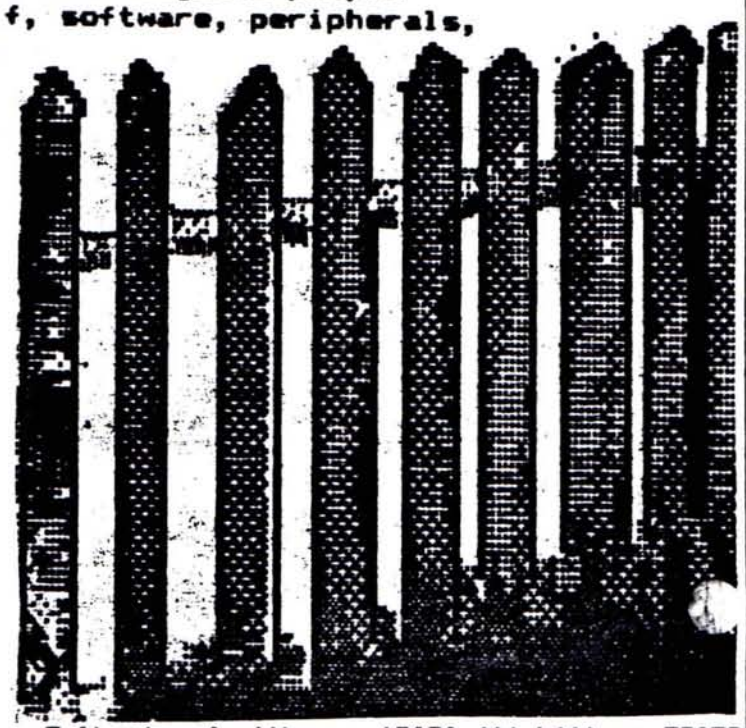
June 86 DOM #4

----------- Utils Disk -----------(8606D)

A:\GOODIES:		
CONVFONT.PRG	DIGIT.PRG!	DISKED.DOC
DISKED.TOS!	DVORAK.TOS!	MAC.TOS!
MACPIC.TOS!	MEMTEST.C	MEMTEST.TOS!
MONO2MED.PRG	NEOCONV.PRG!	POPCORN.PRG!
QWERTY.TOS!	RMD158K.ACC	RMD208K.ACC
ROMTOS2.STW	RTC12.DOC	RTC12.TOS!
SEEKTST.TOS!	SUPERCAL.ACC	TINYTOOL.ACC

The Back Fence by Quinn

This is a new column in the JOURNAL concerned with the ST and all the various goings-on around it. As the above title suggests (rather broadly), much of the content will be the whispers and speculations of more-or-less knowledgable people involved with the ST; the computer itself, software, peripherals, politics and whatnot. A *lot* of whatnot.



There are two primary criteria for the column: it must be informative, and it must be fun. If there comes a conflict between the two, fun will win.

There are some ground rules I will follow. I will not lie. I will not misrepresent. I will not indulge in petty, malicious, maleficent misinformation (even for fun).

This will be a place for good, honest gossip without all of the tedious confirmation, second-source attribution and qualification that gets in the way of reporting the neat stuff. Like I said, fun.

If you get wind of something, let me know about it. Sources will be held in confidence unless you tell me otherwise. Get in touch with me by leaving private mail (for Quinn) on The Village BBS (415-783-5545) or sending hard mail to me care of The JOURNAL. Now on with it.

The big items drawing a respectable amount of speculation these days are the in-process upscale model of the ST and the MS-DOS expansion box. Word of the latter (the MS-DOS box) is shifty. There was some loose talk (even from out of The-House-That-Jack-Built, AKA Atari) that the box would have expansion slots to accept standard PC Boards as well as the 8086 CPU and 8087 math coprocessor and a 5 1/4 inch drive. Now word is that there will be no expansion slots, and no drive and perhaps no provision for adding a drive from the coprocessor board. Not only does that severely limit the usefulness of the box, but it also uses up the DMA port which is also needed for a hard disk. Maybe we can lobby for at least one expansion slot that would take a card with a couple of extra DMA ports. How 'bout it Atari?

Regarding the new high-end 32 bit machine (the one that got some early ink in Byte Magazine last spring) speculation is now more speculative and so more grandiose. Talk is of either a 68020 (16 bit) or 32032 (32 bit) CPU chip with most votes going for the 68020. What *really* gets fun is the talk about two (count 'em, two) 20 Megabyte harddrives built in, the better to support a full UNIX* (trade mark of AT&T Bell Labs) System V implementation. That, a Meg or two of memory with expansion capability, a 1024 x 1024 pixel monitor and all for under (that is *under*) $2,000. If Jack can pull that off, I invite him over to walk across my swimming pool!

More realistically, we might be able to expect a similar box with one drive for under $3,000 by about the middle of next year. That's still a reasonably high-powered UNIX* (see above) workstation at a remarkable price. Hope they can make it.

I think that's about it for this time. The above material is not necessarily the opinion of the SLCC JOURNAL, SLCC, or any other organization or person mentio[illegible] herein. (May not even be my opinion, [illegible] knows?) Drop me a line on the Village with what you know or what you want to know. And I'll see you at the ST-SIG.