Ipso Facto

ISSU 40 May 84

INDEX PAGE

A PUBLICATION OF THE ASSOCIATION OF COMPUTER-CHIP EXPERIMENERS

1983-1984 EXECUTIVE OF THE ASSOCIATION OF COMPUTER CHIP EXPERIMENTERS

President: John Norris 416-239-8567 **Vice-President:** Tony Hill 416-876-4231

Treasurer: Ken Bevis 416-277-2495 **Secretary:** Fred Feaver 416-637-2513

Directors: Bernie Murphy - Fred Pluthero - John Norris - Mike Franklin

Newsletter:

Production Manager: Mike Franklin 416-878-0740

Editors: Fred Feaver
Tony Hill

Publication: Dennis Mildon
John Hanson

Product Mailing: Ed Leslie 416-528-3222 (Publication)

Fred Feaver 416-637-2513 (Boards)

Club Mailing Address:

A.C.E.
c/o Mike Franklin
690 Laurier Avenue
Milton, Ontario
Canada
L9T 4R5
416-878-0740

ARTICLE SUBMISSIONS:

The content of Ipso Facto is voluntarily submitted by Club Members. While ACE assumes no responsibility for errors nor for infringement upon copyright, the Editors verify article content as much as possible. ACE can always use articles, both hardware and software, of any level or type, relating directly to the 1802 or to micro computer components, peripherals, products, etc. Please specify the equipment or support software upon which the article content applies. Articles which are typed are preferred, and are usually printed first. Please send originals, not photocopy material. We will return photocopies of original material if requested.

PUBLICATION POLICY:

The newsletter staff assume no responsibility for article errors nor for infringement upon copyright. The content of all articles will be verified, as much as possible, and limitations listed (i.e. Netronics Basic only, Quest Monitor required, require 16K at 0000-3FFF etc.). The newsletter will be published every other month, commencing in October. Delays may be incurred as a result of loss of staff, postal disruptions, lack of articles, etc. We apologize for such inconvenience - however, they are generally caused by factors beyond the control of the Club.

MEMBERSHIP POLICY:

A membership is contracted on the basis of a Club year - September through the following August. Each member is entitled to, among other privileges of Membership, all six issues of Ipso Facto published during the Club year.

EDITORS CORNER

My apologies for being 3 weeks late with this issue. I wanted to be certain that the CHIP 8 AE article was complete and the program verified. The program is the first major game printed in IPSO for a long time, and it is quite impressive. Thanks Larry, you did quite a job!

Wes Striner also worked long and hard providing a handy index of IPSO articles. It is placed at the back of this issue so you can remove it easily and place it at the front of your IPSO binder for easy reference.

The Seventh annual meeting of ACE took palce or May 8, 1984. John Norris was re elected President, Fred Feaver as Secretary, Ken Bevis as Treasurer. Fred Plethero takes over my duties as Editor on Sept 1, and I go on to Hardware along with Tony Hill. Software adds Mike Smith, Rob Erlich and Dan Thomas, as well as Wayne Bowdish.
Ed Leslie keeps Product mailing in hand, while our trusty Publication committee remains unchanged and true as ever with Dennis Mildon and John Hanson. The new group take over on September 1, 1984.

MEMBERS CORNER

Errata : IF 34 p 22. Two Chip Epromer
On the diagramon page 23, pin 12 of the 2716 is mislabeled as it shouldbe ground. Pin 21 is Vpp.

Comment from Don Stewart of BC. Canada - re ACE adopting a new processor

- Many members do not want to leave the 1802, which is a fine port oriented applications processor.
- Why not use the 1802 as the Input / Output controler and add a 16 bit processor on the buss. That way, the 1802 remains supported, and those who desire better processing can be satisfied too!

For Sale: J Cayer, Box 2034 Hinton Alberta, Canada, phone 403-865-2097
Trade 1 unpopulated ACE backplane ver I, a ACE VDU board and MC6847 and 1372 chips for an unmodified RCA Cosmac VIP.

For Sale: Wes Steiner, 2659 Wildwood Dr., Langly BC Canada V2Y 1G2

Netronics ASCII II Keyboard, Video Display board, metal enclosure and RF modulator complete with all schematics and manuals. Generates 64 or 32 characters upper/lower case by 16 lines on TV or Video monitor. Communicates via RS-232 or 20ma current loop at 300/110 baud. Requires +8 vdc and 8vac (20v p/p) power supply. $100.00

NEW PRODUCTS

The ACE Disk Controller Board ver II is just completing field tests, and should be ready for productinn by the end of May.

The 80 x 24 Video Display Board has completed prototyping, and will be ready for production in late June.

HARDWARE ERRATA - ACE BOARDS 84:04:07

A number of ACE members have writen about the lack of accurate cross referencing of parts lists to drawings on the Front Panel and VDU boards.

Below are listed the values for each board.

<u>Front Panel</u>

R3 to 5 -	470 ohm ¼ watt		Eprom Burner
R2	22k		
R1	10k pot.		
C1	10 mic. tantalum		
C12	10 mic. electro.		
C5	330 " "		
C3,4	.001 " ceramic		
C2	.1 " ceramic		
R7	200 ohm ¼ watt		Hex Pad
R9	5.11k "		
R8	47k "		
C9	22 mic. electro.		
C6	.15 " mylar		
C7,8	1.0 " tantalum		
R14 to 22	470 ohm ¼ watt		Led Display
R23 to 64	130 " "		
R11 - 13	22k " "		Single Step
R9	22k " ½ watt		Clock
R10	100k " ¼ watt		
C10	20 pf ceramic		
C11	20/35pf "	adjustable	

<u>VDU board</u>

Parts list as per page 5 and drawing as per page 6.
add - 22k ¼watt resistor on back of board between pins 13 and 16 of 4025.
add - jumper between S1 (D6) and plate through hole 1" above S1 connected to SIP pin 4 (74C244 pin 7).
cut trace between S1 D6 plate through hole and plate through hole ½" above S1 connected to SIP pin 7 (74C244 pin 7).

<u>Note:</u> I know of no one who has been able to make the VDU board Ver2. work properly with the 1374. My personal experience with the board has been with a direct video output from pin 12 of the 1372, which works fine. Other members who used version 1 reported good results with the 1372 composite output, but it required quite a bit of fiddling and no interference from other electrical signals.

Parts List - ACE Video Display Board, Version 2 (1983.01)

rev. 1983/08/21
rev.2 84:04:07

IC#

1 - 4025
2 - 4068
3 - 74C373
4 - 74C244
5 - 74C10 *HC 10
6 - 74LS139
7 - 74C00 *HC00
8 - 74C20
9 - 74C244
10 - 74C245 *HC245
11 - 4508
12 - 74C244 *HC244
13 - 2016/6116
14 - MC 6847 **Motorola Part
15 - MC1372
16 - 2016/6116
17 - 2016/6116
18 - 2016/6116

19 - MC1374 Optional

* HC CMOS parts or LS TTL parts may be required for high speed operation or to counter significant buss loading

** Motorola part only must be used, other VDU chips such as AMI 68047 have different pin outs.

COMPONENTS

Resistors

1 - 9x22k SIP or 9x22k 1/4w. 5%
3 - 22k
1 - 5k
1 - 5.6k
3 - 240 ohm
2 - 600 ohm
1 - 1k
1 - 15k pot
1 - 10k pot

Optional 1374

3 - 470 ohm
1 - 6.8k
1 - 3.3k
2 - 2.2k
1 - 75 ohm
1 - 560 ohm
1 - 220 ohm
1 - 180k
1 - 30k
1 - 56k

Capacitors

12 - 0.001 bypass
1 - 180 pf
1 - 0.01 uf
1 - 47 pf
1 - 6-30 trimer cap

Optional 1374

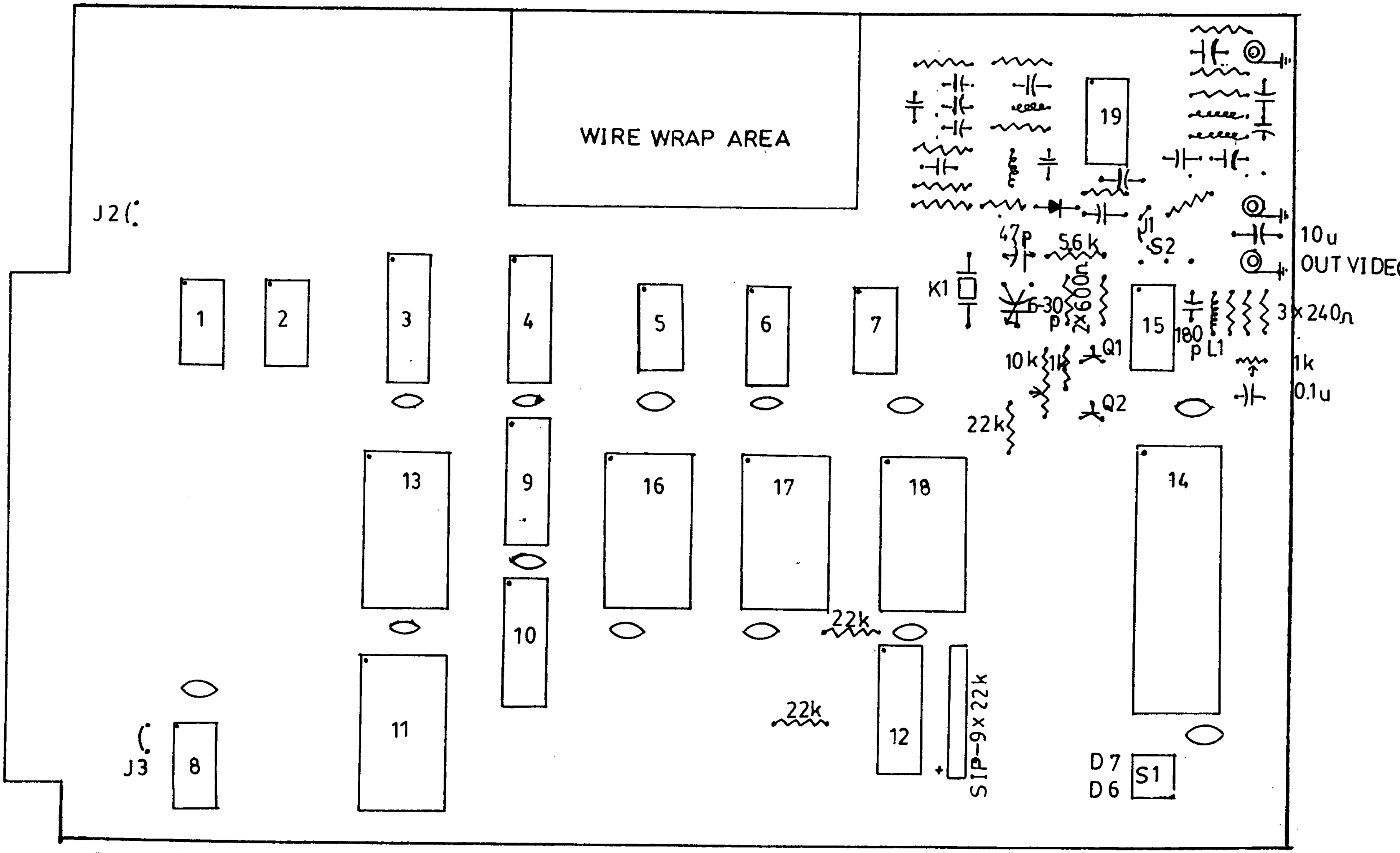
4 - 0.001 uf
1 - 56pf
1 - 120 pf
3 - 47 pf
1 - 10 uf
2 - 22 pf
1 - 0.001 uf

Other Components

1 - 0.1 u henry tuning coil
1 - 3.579 Mhz crystal
2 - PC mount RCA (phono socket)
1 - 4 position Dip Switch
2 - 2N3504

Optional 1374

1 - 1N914 diode
2 - 0.22 u henry coils
2 - inductor coils



BY PASS CAP

ACE VDU Bd ver 2 83-01

INTERFACING THE QUEST 64K DYNAMIC RAM BOARD TO ELF II

*by H.C. Hallaska, 212 N, 70th St. Milwaukee, Wisconsin 53213

This article is in answer to the help article of I.F.#25 and I hope not "TOO LITTLE, TOO LATE". If you are using the Elf II adapter and super - expansion board you are ahead of the game as all the instructions in the manual regarding the super-expansion board are performed as written and will get you to the 50 pin connector from the S-100 buss. Mods follow.

MODIFICATION #1 THE CPU CLOCK

STEP 1 Cut the trace coming from under the 1861 from pin 1 and going upward to plated thru hole near Pin 8 of Elf Buss. Cut just before the hole.

STEP 2 Jumper pins 6 and 8 on the underside of the Elf Buss. NOTE: If you need the CLOCK signal on any accessory boards STEPS 1 and 2 will conflict and another way for signal transfer thru the connectors must be found.

STEP 3 On the solder side of the SUPER-EXPANSIONBOARD connect a wire from S-100 pin 24 to 50 pin connector pin 32.

STEP 4 Verify continuity from 1802 pin 1 and S-100 pin 24.

MODIFICATION #2 THE CPU WAIT STATE CAPABILITY

STEP 5 On the solder side of the Elf II connect a 10K ohm resistor from A-6 pin 2 to A-6 pin 40

STEP 6 No direct equivalent to Elf II.

STEP 7 Remove Jumper J-5 on Elf II.

STEP 8 Jumper pins 10 and 13 on the underside of the 86 pin busson the Elf.

STEP 9 Install a 1N914 diode in place of Jumper J-5 (removed in STEP 7) cathode toward A-13 pins 5 and 6.

STEP 10 On the solder side of the SUPER-EXPANSION BOARD connect a wire between 50 pin connector pin 38 and S-100 pin 72.(Verify continuity from 1802 pin 2 to S-100 pin 72).

MODIFICATION #3 TPA SIGNAL

STEP 11 On the solder side of the super-expansion board connect a wire between U-12 pin 3 and S-100 pin 76.

End of Elf II mods.

Do the steps in Appendix C to assure hardware loading in first 8K, a must for ElfII.

There was an ADDENDUM on a timing problem with the MC3480 and J-1 had a terminal designation mixup. If you didn't get one with your kit Quest would probably send you one on request.

My board has only been up and running for 4 days at the time of this writing but it seems to be OK. No known bugs. I am presently awaiting delivery of Quest Super-Basic, which will give it the "ACID TEST".

Ending this on a note of humor. An addition to Gary Jones Murphy's Laws article of I?F, #37. "The transistor will always blow first to protect the fuse."

HAPPY COMPUTING

64K MEMORY BOARD MODIFICATION TO REPLACE THE 4116 CHIPS WITH 4164 CHIPS

KEN BEVIS

The ACE 64K memory board was designed to use 4116 chips, with a little work you can use the 4164 chips and dispense with the -5 and +12 power supplies.

Since only one row of 4164 chips is required, only one row on the board was modified. In my case I already had 4116's in CAS positions 1 & 2, I decided to modify CAS position 3, that is memory chips 18 - 24.

The +12 volts must be removed from pin 8 and the +5 volts on pin 9 must be moved over to pin 8, all capacitors on pin 9 must be removed. Pin 9 will become an additional address line, pin 1 was not changed at this time but it was planned to remove the -5 volts on completion.

STEPS

1. Remove the +5V from pin 9 at 4 locations on the solder side at chips 23, 21, 19, 17. These are very short foil connections. Pin 9 is also bussed on the chip side of the board and will not test open until all 4 have been . removed.
2. Repeat as in item 1 at chips 24, 22, 20, 18, to clear off pin 8 from the +12v.
3. When pins 8 and 9 are cleared then strap the +5 volts over to pin 9 in at least 2 places, I did it at chips 24, 22, and 20.
4. At this point I plugged in the board and powered up to check pin 8 for +5 and pin 9 that it was floating, stay clear of pin 1 with -5. The 4164 has no internal conn- ection to this pin.
5. Clear the ground or +5 from the following pins, 2 of chip 16, 18 of chip 15. These 2 were straitforward, the next 2 are more difficult. Pin 11 of chip 12 requires cutting the foil above and below and bridging around with a short piece of #30. Pin 3 of chip 15 is no longer connected to ground but is connected to pin 18 under chip 15 and must be opened. If 15 is socketed it may be easy, in my case I had the chip soldered in and solved the problem by drilling a hole up through the board to open the trace.
6. With chip 15 pins 3, 18, chip 12 pin 11 and chip 16 pin 2 all open then proceed with wireing up address lines to pin 9 of the row of memory chips. Connect pin 9 at chip 17 to both chip 16 pin 18 and to chip 12 pin 9. Connect chip 16 pin 2 to chip 15 pin 19 and chip 12 pin 11 to chip 15 pin 2. Now connect chip 15 pin 3 to chip 12 pin 6 and chip 15 pin 18 to chip 16 pin 17.
7. It's now time to power up again and verify that the CPU still runs and all address lines are still function- ing with the additional latches on A6 and A7.
8. I instaled One 4164 chip in the new row and accessed it at address 9000h before plugging in all 8.
9. Since I have EPROM, VIDEO, I/O and small bits of ram in the CO - FFh block I wanted to dissable this section of ram. I cut the trace from chip 10 pin 1 to the CAS3 on the row of memory and installed a 10K pullup to +5 volts. I then installed diodes from CAS1, CAS2 and chip

10 pin 1 to the CAS3 lead. Now remove all 4116 chips.

Don't forget to Test it out with a memory test program and remove any hazzards like the +12 and -5.
There is othor ways of addressing this board if you wanted to install more chips and do bank select, by rearranging the address lines on 4514 decoder you would not require the diodes. A decoding arrangement using the uppermost addresses could select 64K blocks.

COSMIC APPLICATION FOR THE COSMAC

by Steven S. Coles, 22924 76th Av. W. #61, Ednonds, WA. 98020

On page 100 of the March 1983 issue of "Discover" magazine Professor Frank Drake of Cornell University asks eight questions related to the search for extraterrestrial intelligence. In the seventh question a message from a hypothetical extraterrestrial is to be decoded. It is in a binary code and contains 121 bits. As the only positive integer factor of 121 is eleven let's assume the message is intended as an eleven by eleven video image (this might be wrong - who says that every binary word must contain the same number of bits?). Doubling the bits to make it reasonably square on the video screen, then converting to hex code we get:

```
40  00 00 00 00 00 00 00 00
48  00 00 00 00 00 00 00 00
50  00 00 0F C0 FC 00 00 00
58  00 00 C3 0F CC 00 00 00
60  00 00 33 30 CC 00 00 00
68  00 00 0F C0 30 00 00 00
70  00 00 FF FC 0C 00 00 00
78  00 00 0F C0 30 00 00 00
80  00 00 3F F0 C0 00 00 00
88  00 00 F0 3C 30 00 00 00
90  00 00 C0 0C 0C 00 00 00
98  00 00 00 00 30 00 00 00
A0  00 00 CC CC C0 00 00 00
A8  00 00 00 00 00 00 00 00
```

Well look at that!... Oh, yes. To look at that you will need the video graphics software from the July 1977 issue of "Popular Electronics" or volume 1, number 2 of "Questdata" magazine.

AN INDEX OF FORTH ARTICLES

FIG-FORTH ARTICLES FROM IPSO FACTO IN CRONOLOGICAL ORDER

compiled by Gary Jones
Glendale, Arizona

** CORRESPONDENCE by Tom Crawford. IF #20, p 30.
A call for the formation of an 1802 FORTH Special Interest Group.

** FORTH ANYONE by Wayne Bowdish. IF #22, p 51.
Co-ordination of FORTH activities within A.C.E.

** FORTH INTEREST GROUP OVER 2500 MEMBERS IF #22, p 51.
Membership information concerning the FORTH INTEREST GROUP.

** FORTH IS AVAILABLE FOR THE 1802 by Richard Cox. IF #22, p 55.
Announcement of the sale of FORTH code on cassette for the 1802.

** FORTH INTEREST GROUP NATIONAL CONVENTION IF #25, p 7.
An announcement of the 1981 FIG Convention in Santa Clara.

** MOUNTAIN VIEW PRESS IF #25, p 8.
Publications about FORTH available from "the FORTH source".

** EDITORS CORNER - FORTH IF #26, p 3.
Announcement of the availability of "8th", a FORTH derivation, and a progress report (of sorts) on the A.C.E. version of FORTH.

** EDITORS CORNER - FORTH IF 27, p 3.
A report on a FORTH presentation by Tony Hill at the December, 1981 A.C.E. meeting.

** FROM TONY HILL'S NOTEBOOK - FORTH by Tony Hill. IF #28, p 9.
Tony Hill reports the first working version of A.C.E. FORTH.

** EDITOR'S CORNER - FORTH IF #29, p 3.
Announcing the offering of A.C.E. FORTH on cassette.

** FORTH IMPLEMENTATION NOTES - ACE SYSTEMS by Tony Hill. IF #29, p 6-10. Installation instructions and a sample I/O for installing A.C.E. fig-FORTH on an 1802 microcomputer.

** FORTH - A BRIDGE OVER TROUBLED WATERS by L.A. Hart. IF #29, p20-28.
The philosophy of FORTH, along with an introduction to using FORTH.

** CLUB COMMUNIQUE - SOFTWARE IF #29, p 41.
Price and ordering instructions for A.C.E. fig-FORTH on cassette.

** ADDING SIMULATED DISK I/O AND A LINE EDITOR TO FORTH by Ken Mantei. IF #30, p 37-38. Implementation of a FORTH editor and simulated RAM-Disk for 1802 systems.

** FORTH IMPLEMENTATION NOTES II by Tony Hill. IF #31, p 8-9.
Errata and bugs in 1802 fig-FORTH, and notes on a FORTH assembler.

** AN 1802 ASSEMBLER FOR 1802 fig-FORTH by Ken Mantei. IF #31, p 10-11. Instructions and code for developing a FORTH assembler vocabulary.

** EDITOR'S CORNER - SOFTWARE IF 32, p 3. Comments on Club response to the release of 1802 FORTH.

** FORTH IMPLEMENTATION NOTES - 3 by Tony Hill. IF #32, p 8-9. A review of Leo Brodie's "STARTING FORTH", bugs in the "?STACK" word, and notes on 1802 register usage.

** 1802 fig-FORTH "MATCH" AND STRING EDITOR by Ken Mantei. IF #32, p 10-11. Addition of string handling capability to the FORTH editor.

** A LETTER TO THE EDITOR by Fred Hannan. IF #33, p 6-7. A member responds concerning FORTH and "High Tech" articles.

** AN EXTREMELY BIASED OPINION OF FORTH by Steve Nies. IF #34, p 6-7. A discussion of advantages and disadvantages of the FORTH language.

** MORE FORTH PROGRAMS by Tony Hill. IF #34, p 8-10. A disassembler, a memory dump, and fixes for bugs in the double number multiply problem, and the "<" and ">" words.

** USING fig-FORTH WITH SYSTEMS USING INTERRUPTS by Tony Hill. IF #35, p 7. A patch for systems using the 1861 video chip.

** AN 1802 THREADED CODE IMPLEMENTATION by Ed Redman. IF #35, p 8-10. An article about a FORTH-like language.

** AN 1861 TVT FOR FORTH by David Ruske. IF #35, p 31-34. A TVT routine for systems using FORTH with the 1861 video chip.

** THE SECOND ANNUAL 1802 COMPUTER CONFERENCE by Fred Fever. IF #38 (Oct, 83), p 3. Dr. McSolntseff of the Southern Ontario FORTH Interest Group gave a talk on "Programming with FORTH".

** A VLIST FIX FOR FIG-FORTH by Rick Poore. IF #38 (Oct,83), p 15-16. Changing the format of the VLIST word output.

** AN 1802 ASSEMBLER - FORTH STYLE! by Steve Nies. IF #38 (Oct,83), p 19-26. An 1802 FORTH assembler which solves the page boundary branch problem.

** FIG-FORTH EXPANSION by Thomas E. Jones. IF #38 (Oct,83), p 31-34. Using SCRT machine language calls from FORTH, and installing the TCALL routine. Tips on crashing your system.

** FORTH AND THE SMARTERM-80 by Michael Smith. IF #38 (Oct,83), p35-38. An I/O routine and video graphics for FORTH using Netronics' Smarterm-80 terminal.

** MEMBER'S CORNER - A letter from Carlos Qualls. IF #38 (Dec,83), p 4. Carlos asks for help on a cassette based FORTH system.

** FORTH : RIGHT 1802 ASSENBLY CODE by David Horner. IF #38 (Dec,83), p 25. A solution to interrupt problems by changing the "I", "LOOP", and "+LOOP" words.

CHIP-8 AE (ACE EXTENDED)
PIXIE V2.0
VDU V2.0

This article presents an updated version of CHIP-8. Actually, it is two versions, one for the 1861 Pixie chip, and the other for the ACE VDU display board. Both versions are almost completely compatable with each other, and with the original CHIP-8 by RCA.

These new versions feature an expanded instruction set, a choice of display resolutions, and the builtin facility for even further future expansion. Although these versions are intended to reside in memory starting at address Hex 1000, they are relocatable anywhere in memory, provided that they start on a page boundary. Both versions require 1.5K for the interpreter. The VDU version has its display memory already assigned by the hardware, while the Pixie version requires a minimum 1/4K, and a maximum 1K of display memory (depending on the desired display resolution) at any location, starting on a page boundary. For the Pixie, this is, by default, starting on the first page above CHIP-8 AE, which is at Hex 1600 if the interpreter starts at 1000. This default may be changed (more on that later).

It may be wondered why, when the original CHIP-8 required only 1/2K of memory, these updated versions require three times that amount. First, the original "borrowed" a few routines from the monitor in ROM. Second, these updated versions now have about twice as many instructions as the original. Third, the price of memory has fallen quite a bit since the original was written, and thus compact code was not a prime consideration in writing these versions. In fact, speed of executioon took a higher priority in most of the routines. And fourth, these versions now have built-in character generator tables for 63 ASCII characters in addition to the original 16 Hex characters.

If you're interested in writing programs, and are intrigued by what you've read so far, then skip ahead and peruse the Instruction Set (please keep ACE in mind when you come up with that Final, Does-All, End-All Gem). But even if your only interest is to run prewritten CHIP-8 games and such, one of these versions may be for you.

COMPATABILITY

Just how compatable are these versions with the original CHIP-8? All my tests so far indicate that there is very little problem in running prewritten CHIP-8 programs. The biggest problem that has shown up is in CHIP-8 programs which use machine language subroutines. Some of these may run OK, while others will not. It depends on which 1802 registers are being used by the machine language subroutine, and on what the machine language subroutine is performing. Due to the differences in hardware for which these new versions are intended, it was impossible to maintain 100% compatability in this area. Another area where some minor differences show up is in the particular keyboard used on any given system. More on that later. With these two caveats out of the way, I believe I can safely say that most CHIP-8 programs should run completely unmodified on these updated versions.

GETTING IT UP AND RUNNING - VDU VERSION

OUTPUT ROUTINE - just type in the code as presented in this article. Nothing else should be required.

INPUT ROUTINE - The keyboard input routine for the VDU version is intended for a parallel input, ASCII keyboard.. It is located from 1483 to 1495. It tests for a keypressed signal using EF3 (3E at 1489), and it fetches the keyboard data using an INP 7 instruction (6F at 148D).

If you have a serial input keyboard, or otherwise cannot use the routine supplied, you will have to write your own. In this case, the addresses allocated to this routine (1483 to 1489) may not be sufficient. The only thing to do now is to locate your routine on some other page, necessitating the use of a long jump instruction. This will destroy the relocatability of this interpreter, but there's no easy way around it. Put your long jump (C0 __ __) starting at 1489. This will make your routine run with R4 as the Program Couunter and R2 as the Stack Pointer, pointing at the first free location on the stack. Also, DF will be cleared.

Your keyboard input routine must return with the keyboard input byte both in the D register (accumulator) and on the stack. On return, R2 should be unchanged, and pointing at the data on the stack. Also DF should be set if the keypressed signal is true, otherwise DF must be cleared. Additionally, you may want to include a test for a specific ASCII character (such as ESC, Hex 1B), upon which your routine will jump directly to your system Monitor. At any rate, this routine should not wait for a keypressed signal. It should return to the caller by jumping to 1445.

GETTING IT UP AND RUNNING - PIXIE VERSION

OUTPUT ROUTINE - as mentioned earlier, the Display Memory starts by default on the first page above the interpreter program. This should be OK for almost everyone. But, if for any reason, it should be necessary to change this default value, read on. The high byte of the Display Memory start address is determined by a routine located at 15EE to 15F4. This routine looks at its own present location (page 15), adds 1 to it (resulting in 16), then stores the result where the Interupt Routine can find it.

To chaange the default start value, just change the byte at 15F2, which is the amount added to the present location to come up with the high byte start address of Display Memory.

For the Pixie version, the I/O assignments have been written as follows:

```
Turn Pixie On  - 61 (OUT 1) at 14D3
Turn Pixie Off - 62 (OUT 2) at 1381 and 1417
Pixie Status   - is EF1: 3C (BN1) at 15B5 and 15BE
                         34 (B1)  at 15C5
```

If the OUT instructions must be changed to INP instructions, then both the byte PRECEDING and also the byte FOLLOWING the INP instruction should be changed to E2.

INPUT ROUTINE - the input routine supplied for the Pixie version is intended for a Hex keypad using an INP 4 (6C) instruction at 148E to fetch the data and BN4 (3F) at 1491 to test for the Input Key being pressed. This routine also uses an OUT 4 (64) instruction at 148F so that the input data is displayed on the Hex Display. If this routine (located from 1483 to 1496) must be rewritten, and there is insufficient room here for it, you can use up some free space at 1497 to 14D1 inclusive.

SOME GENERAL KEYBOARD NOTES

Due to differences in keyboard hardware, various 1802 systems may react differently to any given CHIP-8 program. This is due generally to the various ways in which the hardware indicates to the software that a key has been pressed. Some ASCII keyboards generate a keypressed signal which is true as long as a key is held pressed, some generate only a short duration strobe signal, and some set a flip-flop which will automatically be reset when the software reads the keyboard (this latter arrangement is ideal for CHIP-8 AE).

On the other hand, most Hex keypads that I have seen on 1802 systems generate a keypressed signal as long as the Input Key, and only the Input Key, is held pressed.

All CHIP-8 AE instructions which access the keyboard have been written to include a check of the keypressed signal. On some systems, depending on your hardware, it may be desireable to turn off this check, in order to improve the playability of some games. For this reason, I include the following chart:

CHIP-8 AE INSTRUCTION	ADDRESS	DATA (CHECK) ENABLED)	DATA (CHECK DISABLED)
EX9E	13AE	3B	38
EXA1	13A4	3B	38
EXA3	13A4	3B	38
EXAD	13AE	3B	38
EXB5	13B8	3B	38

EXB7	13B8	3B	38
EXBF	13C2	3B	38
EXC1	13C2	3B	38

In addition to the differences in keypressed signal generation, there is also the difference in data presented by the keyboard. Obviously, Hex data is not the same as ASCII data. In this case, Hex keypads should have fewer problems with CHIP-8 programs than ASCII keyboards. In fact, most ASCII keyboards will have little problem with CHIP-8 programs which require the pressing of a number key to cause an action to occur on the screen. The problem arises only with the Hex characters A to F. For CHIP-8 programs which require these Hex characters from the keyboard in order to cause an action to occur on the screen, there are two solutions. One is to disassemble the program and change the data expected by the program to some convienient value (key). The other is to use the following ASCII keys to "fool" the software. I leave it up to the user to decide which way to go.

EXPECTED HEX DATA -	A	B	C	D	E	F
EQUIVALENT ASCII KEY -	J	K	L	M	N	O

IN CONCLUSION

From the forgoing, it should be obvious that it was almost impossible to write these new versions of CHIP-8 AE to be 100% compatable with the original. However, very little, if any, modification to existing CHIP-8 programs should be necessary to get them running on your system. In addition, the new expanded instruction set should please those CHIP-8 programmers who, like me, have bumped into the limits of the original CHIP-8.

CHIP-8 AE INSTRUCTION SET (NUMERICALLY)

N	000D	Display Lo-Res Graphics
N	0010	Display Med-Res Graphics
N	0013	Display Hi-Res Graphics
N	0071	Skip if hit on erase
N	0075	Skip if hit on draw
N	007F	Reset Program
P	00DE	Turn on Display
	00E0	Erase Display
	00EE	Return from subroutine
	0MMM	Do machine-language subroutine at 0MMM
	1MMM	Go to 0MMM
	2MMM	Do subroutine at 0MMM
	3XKK	Skip if Vx = KK
	4XKK	Skip if Vx <> KK
	5XY0	Skip if Vx = Vy
N	5XY1	Skip if Vx > Vy
N	5XT2	Skip if timer T = 00

	6XKK	Let Vx = KK
	7XKK	Let Vx = Vx + KK
	8XY0	Let Vx = Vy
	8XY1	Let Vx = Vx OR Vy
	8XY2	Let Vx = Vx AND Vy
N	8XY3	Let Vx = Vx XOR Vy
	8XY4	Let Vx = Vx + Vy
	8XY5	Let Vx = Vx - Vy
N	8XY6	Let Vx = Vy / 2
N	8XY7	Let Vx = Vy - Vx
N	8XT8	Let timer T = Vx
N	8XT9	Let Vx = timer T
N	8XYE	Let Vx = Vy * 2
	9XY0	Skip if Vx <> Vy
N	9XY1	Skip if Vx < Vy
N	9XT2	Skip if timer T <> 00
	AMMM	Let I = 0MMM
	BMMM	Go to 0MMM + V0 of bank 0
	CXKK	Let Vx = random (KK = mask)
	DXYN	Draw N bytes at coordinates Vx, Vy
	EX9E	Skip if Vx = keyboard least significant nibble
	EXA1	Skip if Vx <> keyboard least significant nibble
N	EXA3	Skip if Vx <> keyboard byte
N	EXAD	Skip if Vx = keyboard byte
N	EXB5	Skip if Vx > keyboard least significant nibble
N	EXB7	Skip if Vx > keyboard byte
N	EXBF	Skip if Vx < keyboard least significant nibble
N	EXC1	Skip if Vx < keyboard byte
	FX07	Let Vx = timer (old)
	FX0A	Let Vx = keyboard least significant nibble (MSN = 0)
N	FX0E	Let Vx = keyboard byte
	FX15	Let timer (old) = Vx
	FX18	Let beep-time = Vx
	FX1E	Let I = I + Vx
	FX29	Point I at 5 byte pattern for Vx least significant nibble
N	FX2C	Point I at 5 byte pattern for Vx most significant nibble
	FX33	Let M(I) = 3 decimal digits equivalent of Vx
N	FX50	Clear most significant nibble of Vx
	FX55	Let M(I) = V0 to Vx
	FX65	Let V0 to Vx = M(I)
N	FN74	Let I#0 to I#N = M(I main)
N	FX95	Point I at 7 byte pattern for ASCII character in Vx
N	FNC2	Make variable bank #N the Active Bank
N	FNC6	Save I(main) at I#N
N	FNCA	Restore I(main) from I#N
N	FNCE	Swap I(main) with I#N
N	FXD4	Let I = I - Vx
P	FXFA	Let Display Memory start address high byte = Vx

NOTES: N = New instruction, available on both VDU and Pixie

versions.
P = Specific to Pixie version only!

CHIP-8 AE INSTRUCTION SET (BY FUNCTION)

```
VARIABLES
     6XKK  Let Vx = KK
     7XKK  Let Vx = Vx + KK
     8XY0  Let Vx = Vy
     8XY1  Let Vx = Vx OR Vy (VF of active bank changed)
     8XY2  Let Vx = Vx AND Vy (VF of active bank changed)
     8XY3  Let Vx = Vx XOR Vy (VF of active bank changed)
     8XY4  Let Vx = Vx + Vy (VF of active bank changed)
     8XY5  Let Vx = Vx - Vy (VF of active bank changed)
     8XY6  Let Vx = Vy / 2 (VF of active bank changed)
     8XY7  Let Vx = Vy - Vx (VF of active bank changed)
     8XYE  Let Vx = Vy * 2 (VF of active bank changed)
     CXKK  Let Vx = random (KK = mask)
     FX0A  Let Vx = keyboard lsd (waits)
     FX0E  Let Vx = keyboard byte (waits)
     FX33  Let M(I) = 3 decimal digits equivalent of Vx
     FX50  Clear most significant nibble of Vx
     FX55  Let M(I) = V0 to Vx
     FX65  Let V0 to Vx = M(I)
     FXC2  Make variable bank #N the Active Bank

I POINTER
     AMMM  Let I = 0MMM
     FX1E  Let I = I + Vx
     FX29  Point I at 5 byte pattern for Vx lsd
     FX2C  Point I at 5 byte pattern for Vx msd
     FN74  Let I#0 to I#N = M(I main)
     FX95  Point I at 7 byte pattern for ASCII charater in Vx
     FNC6  Save I at I#N
     FNCA  Restore I from I#N
     FNCE  Swap I with I#N
     FXD4  Let I = I - Vx

TIMERS AND BEEPER
     8XT8  Let timer T = Vx
     8XT9  Let Vx = timer T
     FX07  Let Vx = timer (old)
     FX15  Let timer (old) = Vx
     FX18  Let beep-time = Vx

SKIP
     0071  Skip if hit on erase
     0075  Skip if hit on draw
     3XKK  Skip if Vx = KK
     4XKK  Skip if Vx <> KK
     5XY0  Skip if Vx = Vy
     5XY1  Skip if Vx > Vy
     5XT2  Skip if timer T = 00
```

9XY0	Skip if Vx <> Vy
9XY1	Skip if Vx < Vy
9XT2	Skip if timer T <> 00
EX9E	Skip if Vx = keyboard lsd
EXA1	Skip if Vx <> keyboard lsd
EXA3	Skip if Vx <> keyboard byte
EXAD	Skip if Vx = keyboard byte
EXB5	Skip if Vx > keyboard lsd
EXB7	Skip if Vx > keyboard byte
EXBF	Skip if Vx < keyboard lsd
EXC1	Skip if Vx < keyboard byte

JUMP AND SUBROUTINE

00EE	Return from subroutine
0MMM	Do machine-language subroutine at 0MMM
1MMM	Go to 0MMM
2MMM	Do subroutine at 0MMM (returns with 00EE)
BMMM	Go to 0MMM + V0 of bank 0

DISPLAY AND CONTROL

000D	Display Lo-Res Graphics (default)
0010	Display Med-Res Graphics (default)
0013	Display Hi-Res Graphics (default)
007F	Reset Program
00DE	Turn on Display (Pixie only)
DXYN	Draw N bytes at coordinates Vx, Vy
FXFA	Let Display Memory start address high byte = Vx (Pixie)

CHIP-8 AE INSTRUCTION SET DESCRIPTION

000D DISPLAY LO-RES GRAPHICS

This instruction causes the display to have a resolution of 64 dots across (Hex 00 to 3F) by 32 dots high (Hex 00 to 1F). This is the default resolution for both Pixie and VDU versions, and thus need not be set by a program's initialization phase. This resolution is identical with the original CHIP-8's resolution. In the Pixie version, this instruction turns off the display before changing the resolution, and does not turn it back on again.

0010 DISPLAY MED-RES GRAPHICS

The display has a resolution of 64 dots across (Hex 00 to 3F) and 64 dots high (Hex 00 to 3F) for both Pixie and VDU versions. In the Pixie version, this instruction turns off the display before changing the resolution, and does not turn it back on.

0013 DISPLAY HI-RES GRAPHICS

Due to differences in hardware, this resolution differs between Pixie and VDU versions, and thus a program written to use this resolution on the Pixie will not display properly on the VDU, and vice versa.

PIXIE – The display has a resolution of 64 dots across (Hex 00 to 3F) by 128 dots high (Hex 00 to 7F). The display is turned off before the resolution is changed, and is not turned back on again.

VDU – The display has a resolution of 128 dots across (Hex 00 to 7F) by 64 dots high (Hex 00 to 3F).

0071 SKIP IF HIT ON ERASE

This causes the CHIP-8 instruction following this one to be skipped if, during the erasure of a pattern, it was detected that some other pattern had collided with the one being erased. See DXYN for further details.

0075 SKIP IF HIT ON DRAW

This causes the CHIP-8 instruction following this one to be skipped if, during the drawing of a pattern, it was detected that the pattern being drawn had collided with a previously drawn pattern. See DXYN for further details.

007F RESET PROGRAM

This should be the last instruction executed by a CHIP-8 program. When executed, it waits for a keypressed signal from the keyboard, then restarts the CHIP-8 program from the beginning (at address 0200).

00DE TURN ON DISPLAY

This instruction turns on the Pixie (1861) chip. It should be executed after the following instructions: 000D, 0010, 0013, and FXFA. Alternately, a 00E0 instruction may be executed, as it will automatically turn on the Pixie. This instruction has no effect in the VDU version.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

00E0 ERASE DISPLAY
This instruction erases the display. For the Pixie version, only that memory which is actually being displayed is cleared. Also, in the sPixie version, this instruction automatically turns on the Pixie chip after the display is erased.

00EO RETURN FROM SUBROUTINE
Restores the contents of the Pseudo Program Counter from the stack, so that execution continues from the Pseudo instruction immediately following the last executed CALL instruction (2MMM).

0MMM CALL MACHINE LANGUAGE SUBROUTINE AT 0MMM
This instruction allows dropping from Pseudo-code into machine code, for those rare occasions when Pseudo-code does not perform the job required, or perform it fast enough. When executed, a machine language routine starting at address 0MMM will be executed, with R3 as the Program Counter and R2 as the Stack Pointer, pointing to the first free location on the stack. When the machine language routine is finished, it should return tto the instruction following the last executed 0MMM instuction by using a D4 instruction. Registers which are available for use by the machine language routine are:
NO NEED TO SAVE AND RESTORE - R3, R6.0, R7.0, R9.0, RB.0, RC.0, RD, RE, and RF.
MUST BE SAVED BEFORE USING, AND RESTORED WHHEN FINISHED - R4, R5, R6.1, R7.1, R9.1, RA, RB.1.
Any registers not mentioned must NOT be used under any circumstances, under penalty of your program crashing!

1MMM GOTO 0MMM
As the name implies, this is a jump instruction.

2MMM DO SUBROUTINE AT 0MMM
Causes the present contents of the Pseudo Program Counter to be saved on the stack, and then the PPC is reloaded with 0MMM, causing execution to continue from 0MMM.

3XKK SKIP IF VX = KK
Causes the next pseudo instruction to be skipped over if variable X of the presently active variable bank is equal to KK.

4XKK SKIP IF VX <> KK
Causes the next pseudo instruction to be skipped over if variable X of the presently active variable bank is not equal to KK.

5XY0 SKIP IF VX = VY
Causes the next Pseudo Instruction to be skipped over if variable X is equal to variable Y, both being within the presently active variable bank.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

5XY1 SKIP IF VX > VY
Skips the next Pseudo Instruction if variable X is greater than variable Y, both being within the presently active variable bank.

5XT2 SKIP IF TIMER T = 00
Checks one of the 16 timers specified by T. If the timer has timed out (contains 00), the next Pseudo Instruction will be skipped.

6XKK LET VX = KK
The variable specified by X, of the presently active variable bank, is assigned the value specified by KK.

7XKK LET VX = VX + KK
Adds the value KK to the contents of variable X within the presently active variable bank.

8XY0 LET VX = VY
Moves the value contained within variable Y to the variable X, both variables being within the presently active variable bank.

8XY1 LET VX = VX OR VY
Logically OR's the contents of variables X and Y, and puts the result in variable X. Variable F is changed. All variables are withing the presently active variable bank.

8XY2 LET VX = VX AND VY
Logically AND's the contents of variables X and Y, and puts the result in variable X. Variable F is changed. All variables are within the presently active variable bank.

8XY3 LET VX = VX XOR VY
Logically XOR's the contents of variables X and Y, and puts the result in variable X. Variable F is changed. All variables are within the presently active variable bank.

8XY4 LET VX = VX + VY
Adds the contents of variables X and Y, and puts the result in variable X. Variable F contains the carry information: 00 = no carry, 01 = carry. All variables are in the presently active variable bank.

8XY5 LET VX = VX - VY
Subtracts the contents of variable Y from variable X, and puts the result in variable X. Variable F contains the borrow information: 00 = borrow, 01 = no borrow. All variables are in the presently active variable bank.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

8XY6 LET VX = VX / 2
Divides the contents of variable Y by 2 (actually, does a shift-right), and puts the result in variable X. The lsb of Vy is stored in the lsb position of variable F (as Hex 00 or 01). All variables are in the presently active variable bank.

8XY7 LET VX = VY - VX
Subtracts the contents of variable X from variable Y, and puts the result in variable X. Variable F contains the borrow information (00 = borrow, 01 = no borrow). All variables are within the presently active variable bank.

8XT8 LET TIMER T = VX
Moves the contents of variable X to one of the 16 timers specified by T. Variable X is in the presently active variable bank. All timers are automatically decremented by approximately 60 couunts per second, until they reach a value of 00. Thus an initial value of FF provides a time of slightly more than 4 seconds.

8XT9 LET VX = TIMER T
Moves the contents of one of the 16 timers, specified by T, into variable X of the presently active variable bank.

8XYE LET VX = VY * 2
Multiplies the contents of variable Y by 2 (actually, does a left-shift), and puts the result in variable X. The msb of variable Y is put in the lsb position of variable F (as Hex 00 or 01). All variables are in the presently active variable bank.

9XY0 SKIP IF VX <> VY
Skips the next Pseudo Instruction if variable X is not equal to variable Y, both being within the presently active variable bank.

9XY1 SKIP IF VX < VY
Skips the next Pseudo Instruction if variable X is less than variable Y, both being within the presently active variable bank.

9XT2 SKIP IF TIMER T <> 00
Checks the contents of one of the 16 timers, specified by T. If it has not timed out (contents not equall to 00), the next Pseudo Instruction will be skipped.

AMMM LET I = 0MMM
Assigns the value specified by MMM, with an appended leading 0, to the I Pointer.

BMMM GOTO 0MMM + V0 OF BANK 0
Adds the contents of variable 0 of bank 0 to the value 0MMM. The result is put in the Pseudo Program Counter, so that execution will continue from that point.

CXKK LET VX = RANNDOM (KK = MASK)
A random number (Hex 00 to FF) is logically ANDed with the value KK. The result is then put into variable X of the presently active variable bank.

DXYN DRAW N BYTES AT COORDINATES VX, VY
This instruction draws N bytes (pointed to by the I Pointer) on the screen at an X coordinate specified by variable X (horizontal direction) and a Y coordinate specified by variable Y (vertical direction). The I Pointer is left unchanged. The drawing is done using an exclusive-or process, so that DXYN may be used to both draw and to erase a pattern (if the erasing is done with the same X and Y coordinates). This routine takes note of two types of collisions that may occur between moving patterns on the screen. If a pattern is being drawn, if may land on top of a previously drawn pattern. This is called a Draw-Hit. But if a pattern is being erased, it's possible that another pattern had landed on top of this one between the time it was originally drawn and now. This is called an Erase-Hit. This instruction detects both types of hits, and sets two flags accordingly. But take note:- the Erase Flag will always be set after a DRAW operation, and the Draw Flag will always be set after an ERRASE operation. Therefore, 0071 (Skip if Hit on Erase) should only be used after DXYN has erased a pattern, and 0075 (Skip if Hit on Draw) should only be used after DXYN has drawn a pattern. Both flags, however, remain valid (unchanged) up until the next time that DXYN is used. Also, to maintain compatability with the original CHIP-8, variable F of variable bank 0 is set on a Draw-Hit, ie: Hex 00 means there was no collision, while Hex 01 means there was a collision.

EX9E SKIP IF VX = KEYBOARD LSD
This instruction fetches a byte from the keyboard (on the fly - no waiting around) and compares the least significant digit (nibble) to the contents of variable X of the presently active bank. If they are equal, the next Pseudo Instruction is skipped over.

EXA1 SKIP IF VX <> DD KEYBOARD LSD
This instruction also feetches a byte from the keyboard and compares the least significant digit to the contents of variable X. If they are different, the next Pseudo Instruction is skipped over.

EXA3 SKIP IF VX <> KEYBOARD BYTE
As above, except that the whole byte from the keyboard is used.

EXAD SKIP IF VX = KEYBOARD BYTE
Identical to EX9E, except that the whole byte from the keyboard is used.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

EXB5 SKIP IF VX > KEYBOARD LSD
Again, fetches a byte from the keyboard on the fly, clears the high digit (nibble), then compares it to the value in variable X of the presenttly selected bank. If Vx is greater, the next Pseudo Instruction is skipped.

EXB7 SKIP IF VX > KEYBOARD BYTE
Identical to EXB5, except the high digit of the keyboard byte is included in the comparison.

EXBF SKIP IF VX < KEYBOARD LSD
Fetches a byte from the keyboard on the fly, clears the high digit, then compares it to the value in variable X of the presently active bank. If Vx is lesser, the next Pseudo Instruction will be skipped.

EXC1 SKIP IF VX < KEYBOARD BYTE
Identical to EXBF, except the high digit of the keyboard byte is included in the comparison.

FX07 LET VX = TIMER (OLD)
Assigns the present value of the (old) timer to variable X of the presently active bank. This timer is separate from the other 16 (which use 8XT8, 8XT9, 5XT2, and 9XT2). Although this timer (and this instruction) is now probably redundant, it was included to maintain compatability with the original CHIP-8.

FX0A LET VX = KEYBOARD LSD
This instruction waits for a keypressed signal, then inputs the least significant nibble and puts it in variable X of the presently active bank.

FX0E LET VX = KEYBOARD BYTE
Same as above, but inputs the whole keyboard byte.

FX15 LET TIMER (OLD) = VX
Takes the value in variable X of the present bank and puts it in the (old) timer. As with FX07, this instruction was included to maintain compatability with the original CHIP-8. This counter is automatically decremented by approximately 60 counts per second until it reaches 00.

FX18 LET BEEP-TIME = VX
Takes the value held in variable X of the presently active bank, and puts it in the beep-timer. This causes the beeper to produce a tone (sort of) until the beep-timer counts down to 00, which it does at the rate of approximately 60 counts per second.

FX1E LET I = I + VX
Adds the value found in variable X of the presently active bank to the contents of the I Pointer. A carry to the high byte of the I Pointer is actioned.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

FX29 POINT I AT 5 BYTE PATTERN FOR VX LSD
The least significant digit of variable X of the presently active bank is used to index into a (self-contained) table of patterns. The I Pointer is then pointed at the appropriate pattern, so that a DXYN instruction will cause the LSD of Vx to be displayed. The pattern contains 5 bytes.

FX2C POINT I AT 5 BYTE PATTERN FOR VX MSD
Identical to FX29, except that the most significant digit of Vx is used.

FX33 LET M(I) = 3 DECIMAL DIGITS EQUIVALENT OF VX
The Hex value contained in variable X of the presently active bank is converted to 3 decimal digits which are then stored in 3 consecutive memory locations pointed to by the I Pointer. For instance, if Vx contains the value FE, then after this instruction is executed,

```
M(I)   contains 02
M(I+1) contains 05
M(I+2) contains 04
```

since Hex FE is equivalent to Decimal 254. The I Pointer is left unchanged after this instruction is executed.

FX50 CLEAR UPPER NIBBLE OF VX
This instruction clears (sets to 0) the most significant nibble of variable X of the presently active bank.

FX55 LET M(I) = VO TO VX
Transfers the contents of variable 0 to variable X inclusive to memory pointed to by the I Pointer. After this instruction is executed, the I Pointer contains its original value plus X plus 1.

FX65 LET VO TO VX = M(I)
Transfers the contents of memory locations pointed to by the I Pointer to variable 0 through variable X inclusive. After this instruction is executed, the I Pointer contains its original value plus X plus 1.

FN74 LET I#O TO I#N = M(I MAIN)
Transfers the contents of memory locations pointed to by the I Pointer to the I storage locations #0 through #N inclusive. After this instruction is executed, the I Pointer contains its original value plus 2X plus 1. This instruction is intended to provide a quick and convenient method of initializing the I storage locations.

FX95 POINT I AT 7 BYTE PATTERN OF ASCII CHARACTER IN VX
The ASCII character held in variable X of the presently active bank is used to index into a (self contained) table of patterns. The I Pointer is then pointed at the appropriate pattern, so that a DXYN will display that character. The pattern contains 7 bytes.

CHIP-8 AE INSTRUCTION SET DESCRIPTION

FNC2 MAKE VARIABLE BANK #N THE ACTIVE BANK
There are 16 banks of variables, each containing 16 variables. Only one bank of 16 can be active at a time. This instruction determines which bank is active. Note that variable bank #0 is active by default when the program first starts, and until this instruction is executed. Both the variables and the banks are numbered from 0 to F.

FNC6 SAVE I(MAIN) AT I#N
The contents of the I Pointer are saved in I Storage Location #N. There are 16 storage locations numbered 0 to F. The I Pointer is not changed by this instruction.

FNCA RESTORE I(MAIN) FROM I#N
The contents of I Storage Location #N are transfered to the I Pointer.

FNCE SWAP I(MAIN) WITH I#N
The contents of the I Pointer and I Storage Location #N are swapped.

FXD4 LET I = I - VX
The contents of variable X of thee presently selected bank is subtracted from the value contained in the I Pointer. A borrow from the high byte of the I Pointer is actioned.

FXFA LET DISPLAY MEEMORY START ADDRESS HI BYTE = VX
This instruction is applicable only to the Pixie version, and has no effect on the VDU version. It sets the high byte of the start of Display Memory equal to the contents of variable X of the presently selected bank. The low byte of the Display Memory start address is always 00, so that Display Memory always starts on a page boundary. If this instruction is not executed, the Display Memory will start, by default, at the first page above the CHIP-8 AE Interpreter Program. Note that the Pixie chip is turned off before this address is changed, and is not turned back on again.

CHIP 8 AE code

Common code - 1000 to 13FF.

```
1000   93 FC 03 BB FC 01 B4 F8 01 B6 B7 F8 7F A2 F8 00
1010   A4 F8 CD A9 F8 00 B2 B8 A8 B9 59 19 99 32 1A 29
1020   D4 83 3B 20 D4 83 33 24 93 30 01 E2 D4 5F E2 15
1030   95 73 85 73 25 05 A5 9E B5 D4 24 45 F3 3A 41 15
1040   15 D4 24 45 F3 3A 3F D4 24 F8 4A 30 53 F8 53 30
1050   53 F8 64 AB D4 4D F8 0F 30 5C F8 07 F2 52 8B F4
1060   E6 AB 0B A3 07 30 3C 07 F7 3B 3F D4 24 87 F9 F0
1070   A9 09 30 3D D4 24 E2 E2 D4 24 E2 E2 D4 24 E2 E2
1080   D4 24 E2 E2 D4 24 E2 E2 45 F4 38 45 56 D4 24 07
1090   30 8C D4 66 03 C9 87 F9 F0 A9 06 59 D4 24 87 F9
10A0   F0 A9 09 30 8C D4 24 E2 E2 D4 24 E2 E2 D4 24 E2
10B0   E2 D4 24 E2 E2 D4 24 E2 E2 07 30 44 07 F5 30 69
10C0   87 F9 F0 A9 09 30 45 D4 24 E2 E2 D4 24 E2 E2 D4
10D0   24 E2 E2 D4 24 E2 E2 D4 24 E2 E2 45 AA 9E BA D4
10E0   24 92 A6 45 F4 A5 9E 7C 00 30 3B 9C 56 45 F2 30
10F0   8C F8 72 AB D4 66 05 00 D4 66 03 96 E6 E6 E6 45

1100   A3 56 38 B8 D4 24 E2 98 30 01 F8 0F 30 10 F8 FF
1110   AF D4 66 02 8A 06 30 03 06 A8 D4 24 E6 E6 8A F4
1120   AA 9A 7C 00 BA D4 24 E2 E2 06 30 88 06 F6 F6 F6
1130   F6 30 8A 06 52 F8 6D AB EB 92 AF 02 38 1F F7 33
1140   3D F4 52 8F 5A 1A 1B 0B 3A 39 02 5A 2A 2A D4 24
1150   06 FA 0F 30 01 86 52 E2 FA F0 A6 46 5A 1A 86 F5
1160   33 5B D4 24 E2 86 52 E2 FA F0 A6 4A 56 16 86 F5
1170   33 6B D4 24 86 FA 0F AF F8 D0 A9 38 2F 4A 59 19
1180   4A 59 19 8F 3A 7C D4 24 FA 0F FF 0A 3B 90 FC 07
1190   FC 3A BF 30 98 92 BF 06 FE FE FE AE 9F 7E BF 93
11A0   FC 01 BE 92 BA F8 C2 A9 AA 9F F6 4E 33 B2 FE FE
11B0   FE FE FA F0 59 19 89 FB CA 3A A9 9F F6 32 C0 1A
11C0   D4 24 D4 66 02 00 D4 66 02 0B D4 66 02 1C D4 66
11D0   02 26 D4 66 02 76 D4 24 E2 E2 D4 24 E2 E2 D4 24
11E0   E2 E2 D4 24 E2 E2 D4 24 E2 E2 D4 24 E2 E2 D4 24
11F0   E2 E2 D4 24 E2 E2 D4 24 E2 E2 00 00 00 00 00 00

1200   00 00 00 00 00 00 00 00 00 26 29 2F 09 29 00 00
1210   50 5E 05 07 05 0E 00 00 00 67 F8 68 F8 67 00 00
1220   60 7E 85 65 15 EE 60 00 00 9F 38 6E C8 9F 00 00
1230   00 0F 08 0E 08 08 00 00 40 47 08 0B 09 07 00 00
1240   20 49 49 4F 49 49 20 00 40 2E 24 24 24 2E 40 00
1250   50 21 51 01 09 06 00 00 00 09 2A 7C 2A 09 00 00
1260   00 08 08 08 48 4E 80 00 00 09 0F FF 09 09 00 00
1270   00 09 0D 0F 0B 49 00 00 10 16 29 29 49 46 80 00
1280   00 FF 99 9F 98 F8 00 00 00 26 69 29 2B 76 00 00
1290   00 EF 19 6F 8A F9 00 00 00 E7 18 76 11 EE 00 00
12A0   00 AF A4 F4 24 24 00 00 00 F9 89 F9 19 F6 00 00
12B0   00 FA 8A FA 9A F4 00 00 00 F9 99 1F 1F 19 00 00
12C0   00 69 99 66 99 69 00 00 00 FA 9A F4 14 F4 00 00
12D0   00 07 41 02 44 07 00 00 0E 08 48 08 48 48 8E 00
12E0   08 28 44 84 42 22 01 00 0E 02 F2 02 F2 02 0E 00
12F0   06 49 20 10 20 40 00 00 00 60 90 20 00 20 0F 00
```

```
1300   F8 CF A9 86 FE FE FE FE 59 D4 24 F8 15 AF 9E FE
1310   FC D0 A9 8F A3 9A 59 19 8A 59 D4 24 F8 20 30 0D
1320   49 BA 09 AA D4 24 F8 2A 30 0D 9A 52 09 BA 02 59
1330   19 8A 52 09 AA 02 59 D4 24 23 2B 35 2E 3B 43 49
1340   8B 88 4D 51 DB E1 EB F1 F8 FF 5A 64 67 6D 74 78
1350   7C 80 84 56 8F 92 92 92 92 92 92 92 96 9E A5 A9
1360   AD B1 92 B5 5A B9 BC C0 C7 CB CF D3 D7 64 0A 00
1370   01 AB 02 B9 04 C7 E6 8A F7 AA 9A 7F 00 BA D4 24
1380   00 00 00 00 00 00 00 00 00 00 D4 83 3B 8A D4 83
1390   33 8E 8F F2 30 DB D4 83 E6 FA 0F AF 45 A3 8F 30
13A0   AE 8F 38 02 3B A9 F3 32 AB 15 15 D4 24 02 3B AB
13B0   F3 32 A9 D4 24 8F 38 02 3B 7D F7 3B A9 D4 24 8F
13C0   38 02 3B C7 F5 3B A9 D4 24 22 F8 D3 52 22 9F F9
13D0   F0 52 07 D2 56 86 F9 0F A6 92 7E 56 D4 24 23 23
13E0   23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
13F0   23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23
```

<u>CHIP 8 AE Driver</u> - VDU version

```
1400   94 FC 00 B1 FC 01 B5 F8 96 A1 F8 FA A5 F8 03 C8
1410   F8 01 C8 F8 00 52 F8 CC A9 02 59 F8 FF BF F8 FF
1420   AF F8 32 5F 79 D1 00 F8 CF A9 E9 05 3A 31 15 45
1430   A4 FA 0F BE F1 A6 45 F6 F6 F6 F6 FC 3A AB 0B A3
1440   94 FF 04 B3 E6 D3 43 A4 E2 E2 E2 E2 E2 05 52 FA
1450   0F BF 45 F6 F6 F6 F6 E9 F1 A7 E2 4B A3 30 45 45
1460   A3 9E B3 D3 30 24 22 E2 43 52 03 A3 93 F4 12 30
1470   43 F8 CE 30 77 F8 CD A9 09 32 24 15 15 30 24 F8
1480   20 30 3F 79 D1 00 FC 00 3E 8C FF 00 E2 6F FB 18
1490   C2 FE 00 02 30 45 89 52 F8 CA A9 09 34 A4 72 3A
14A0   C3 59 30 C5 3A BE 96 59 F8 F0 A9 09 CE FF 01 59
14B0   19 99 32 AB 29 98 CE FF 01 B8 88 32 BE 28 72 32
14C0   C5 34 C1 3C C3 88 32 CB CD 7B 38 7A 9C FC 01 BC
14D0   02 A9 E2 78 70 30 96 00 00 00 00 00 00 00 00 00
14E0   79 D1 01 F8 FF AF F8 E3 BF EF 92 73 30 F5 12 42
14F0   A5 02 B5 30 24 9F FB DF 3A EA 30 24 E2 D5 30 24

1500   D4 4D 9F 32 FA AF 9A 73 8A 73 06 FA 07 BD F8 AC
1510   A9 92 AE 4A BE 32 25 9D 32 25 AD 9E F6 BE 8E 76
1520   AE 2D 8D 3A 1B 9E 59 19 8E 59 19 2F 8F 3A 11 F8
1530   CC A9 09 AA 32 6F 79 D1 00 F8 AC A9 F8 8E AE 92
1540   BE 9F FE AF F8 10 AC 09 3A 52 5E 1E 30 65 09 FE
1550   59 C8 09 FE 8D 7E AD 9D 7E BD 2C 8C F6 33 4E 3A
1560   52 9D 5E 1E 8D 5E 1E 19 2F 8F 3A 44 F8 8E C8 F8
1570   AC A9 8A F6 06 FA F8 3B 7A FE FA 78 F6 F6 F6 52
1580   8A F6 F6 07 C7 FE FE FE FE AE 92 BD AD 7E FE BE
1590   8E FE F1 AE 9E 7C E0 BE 79 D1 01 E9 8A C4 F6 F8
15A0   04 CF F8 02 AC 8C AF 0E F3 5E F2 32 AE BD F3 32
15B0   B2 AD 19 1E 2F 8F 3A A7 8C AF 2E 2F 8F 3A BA 8E
15C0   FC 10 AE 9E 7C 00 BE FD E3 33 CE F8 E0 BE 8A F6
15D0   32 E0 8E FA 10 32 E0 8C AF 29 2F 8F 3A D9 30 A5
15E0   9F FF 01 BF 3A A5 F8 CD A9 F8 0F A6 8D 59 32 F1
15F0   96 56 9D 19 59 12 42 AA 02 BA D4 24 00 E0 12 00
```

CHIP 8 AE Driver - 1861 version

```
1400   94 FC 01 B1 FC 00 E5 F8 9C A1 F8 EC A5 F8 70 C8
1410   F8 72 C8 F8 74 AB E4 62 00 F8 CA A9 4B 59 19 19
1420   0B 59 E2 E2 88 32 2B 31 2B 7B 38 7A F8 CF A9 E9
1430   05 3A 36 15 45 A4 FA 0F BE F1 A6 45 F6 F6 F6 F6
1440   FC 3A AB 0B A3 94 FF 04 B3 E6 D3 43 A4 05 52 FA
1450   0F BF 45 F6 F6 F6 F6 E9 F1 A7 E2 4B A3 30 4A 45
1460   A3 9E B3 D3 30 24 22 E2 43 52 03 A3 93 F4 12 30
1470   48 F8 CE 30 77 F8 CD A9 09 32 24 15 15 30 24 F8
1480   20 30 44 88 32 8A 31 8A 7B 38 7A FC 00 E2 6C 64
1490   22 3F 4A FF 00 30 4A 24 24 24 24 24 24 24 24 24
14A0   24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
14B0   24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
14C0   24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
14D0   24 23 E4 61 00 30 24 92 5E 1E 2F 9F 30 D7 30 D2
14E0   F8 CA A9 49 BF F8 FF AF 09 BE 92 AE 30 DB 12 42
14F0   A5 02 B5 30 24 24 24 24 24 24 24 23 E2 D5 30 24

1500   D4 4D 9F 32 97 AF 9A 73 8A 73 06 FA 07 BD F8 AC
1510   A9 92 AE 4A BE 32 25 9D 32 25 AD 9E F6 BE 8E 76
1520   AE 2D 8D 3A 1B 9E 59 19 8E 59 19 2F 8F 3A 11 06
1530   FA 3F F6 F6 F6 52 92 BE BD AD 07 FE FE AE 9E 7E
1540   BE 8E FE F1 AE F8 CB A9 E9 9E 7E F4 BE 29 49 F4
1550   AC 09 A7 F8 AC A9 E2 9E 52 8C F5 3B 5F 87 BE 96
1560   AF E9 38 1E 0E F3 5E F2 32 6B BD F3 32 6F 1D 19
1570   2F 8F 32 63 9F 32 82 2E 8E FC 08 AE 9E 7C 00 BE
1580   30 56 F8 CD A9 F8 0F A6 8D FC FF 92 7E 56 59 19
1590   9D 59 12 42 AA 02 BA D4 24 23 78 70 C4 22 73 89
15A0   73 F8 CB A9 49 B0 92 A0 49 A1 E2 80 E2 20 A0 E2
15B0   20 A0 E2 20 A0 3C AA 30 C7 80 E2 20 A0 E2 3C B9
15C0   80 E2 20 A0 E2 34 C0 76 52 9C FC 01 BC 98 32 D3
15D0   FF 01 B8 88 32 D7 28 F8 F0 A9 09 32 E0 FF 01 59
15E0   19 99 32 DA 29 42 7E 42 A9 02 30 9A 00 FC 92 A6
15F0   95 FC 01 56 D4 F0 FA 00 E0 12 00 23 23 23 23 21
```

Editors Note:

The above program and drivers have been verified against a club ACE VDU ver2 board and on a home built 1861 circuit.

The FX0A command will retain only bit count equivolent to Hex 0 - 9, thus certain games or programs requiring an input of Hex A - F will require modification, or will not run properly.

The above program, with both drivers, complete with commented listing is available from ACE on a cassette (Netronics CassetteI/O) for $ 10.00.

by - Wes Steiner, 2659 Wildwood Dr., Langly B.C. Canada, V2Y 1G2

IPSO FACTO INDEX OF ARTICLES
YEARS 1 TO 7 ISSUES 1 TO 38
INDEXED BY KEYWORD

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
ASSEMBLER	I	A CROSS ASSEMBLER - WHAT'S IT MAD AT?	W BOWDISH	1	2	9
ASSEMBLER	I	NOTES ON THE DON STEVEN'S NOTABLE ASSEMBLER	D THORNTON	6	36	33
ASSEMBLER	S	AN 1802 EDITOR/ASSEMBLER	G MILLAR	2	12	60
ASSEMBLER	S	1802 PASCAL CROSS ASSEMBLER	R BLESSING	3	17	4
ASSEMBLER	S	NEW MNEMONICS FOR THE NETRONICS ASSEMBLER	D SCHULER	5	28	23
ASSEMBLER	S	AN 1802 ASSEMBLER - FORTH STYLE	S NIES	7	37	19
		--				
AUDIO	C	MUSIC AND MICROS	E TEKATCH	1	1	5
AUDIO	C	THE 1802 MUSIC MACHINE	C WILLIAMS	1	5	36
AUDIO	C	ELECTRONIC METRONOME	R EDWARDS	2	7	7
AUDIO	C	PROGRAMMABLE SOUND GENERATOR	H SHANKO	4	20	25
AUDIO	C	PROGRAM YOUR SOUNDS	E SHAFFER	4	24	41
AUDIO	H	TONE GENERATOR	M COYNE	4	24	32
AUDIO	S	A FINE RESOLUTION AUDIO OSCILLATOR PROGRAM	R EDWARDS	1	5	20
AUDIO	S	VARIATIONS ON A THEME	D HERSKER	2	8	28
AUDIO	S	COMPUTER CHRISTMAS MUSIC	C AIRHART	2	8	45
AUDIO	S	4 BYTE DELIGHT	E JACKSON	3	14	14
		--				
BASIC	S	ENHANCEMENTS TO HANNAN'S TEXT EDITOR		7	38	9
BASIC	S	16 BYTE WIDE HEX DUMP	G JONES	7	38	15
BASIC	S	WORD PROCESSOR II	T NERY	7	38	17
		--				
CASSETTE	C	AN APPLICATION OF MEMORY-MAPPED I/O	T JONES	3	13	25
CASSETTE	H	CASSETTES AND COMPUTERS	T CRAWFORD	1	3	23
CASSETTE	H	CASSETTE I/O MONITOR	R THORNTON	3	15	35
CASSETTE	H	IMPROVED TAPE CONTROLLER	D SCHULER	4	19	25
CASSETTE	H	ANOTHER TAPE CONTROLLER	W SCHULTZ	5	28	33
CASSETTE	H	HOMEBREW ELF ENHANCEMENTS (LED INDICATOR)	A BOISVERT	6	35	27
CASSETTE	I	CUTS - COMPUTER USER TAPE SYSTEM	R MARSH	1	3	19
CASSETTE	I	A STANDARD FORMAT FOR CASSETTE DATA	W BOWDISH	1	3	31
CASSETTE	I	CERTIFYING AUDIO TAPE FOR DIGITAL USE		1	5	21
CASSETTE	I	MAGNETIC TAPE DATA RECORDING	R SMITH	1	5	24
CASSETTE	I	A SOFTWARE STANDARD FOR KANSAS CITY FORMAT TAPES	B MURPHY	1	6	5
CASSETTE	I	CASSETTE LOAD VOLUME SENSITIVE?	B ECKEL	4	19	29
CASSETTE	I	ELF II CASSETTE I/O	K MANTEI	4	20	37
CASSETTE	I	NETRONICS MONITOR CASSETTE PROBLEMS	T JONES	5	25	11
CASSETTE	S	RCA 1802 - KC STANDARD CASSETTE INTERFACE TEST ROUTINE	A DUNLOP	1	3	29

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
CASSETTE	S	A DIFFERENT CASSETTE I/O ROUTINE	R EDWARDS	1	6	61
CASSETTE	S	SOFTWARE FOR THE "IPSO STANDARD" KANSAS CITY TAPE INTE	B MURPHY	2	8	37
CASSETTE	S	RUNNING THE "IPSO FACTO STANDARD" ON AN ELF	B MURPHY	2	9	24
CASSETTE	S	KEY-IN LOADER FOR ELF-II FORMAT TAPES	T JONES	2	9	34
CASSETTE	S	CASSETTE "FILE COUNTER" SYSTEM	T CRAWFORD	3	13	7
CASSETTE	S	MINI TAPE LOADER	R THORNTON	3	15	38
CASSETTE	S	NETRONICS COMPATIBLE TAPE LOAD PROGRAM	M FRANKLIN	6	31	41
CASSETTE	S	CASSETTE COPY PROGRAM	A BOISVERT	6	34	31
		--				
CLOCK	C	A HARDWARE CLOCK FOR THE 1802	J SWOFFORD	5	27	26
CLOCK	C	THE WD2412 TIME OF DAY CLOCK	D SCHULER	6	36	23
CLOCK	H	A SOFTWARE CLOCK LOCK	K BEVIS/P MUIR	6	34	32
CLOCK	S	REAL TIME CLOCK AND PROGRAM	M FRANKLIN	6	32	35
CLOCK	S	ACE HARDWARE DRIVERS FOR SYMON	M FRANKLIN	6	34	33
CLOCK	S	TIME SET REAL TIME CLOCK DRIVER	M FRANKLIN	6	34	34
		--				
CLUB NEWS	I	THE STANDARDIZED 1802	S NIES	3	17	17
CLUB NEWS	I	HARDWARE PROJECTS AND YOU		4	19	30
CLUB NEWS	I	INTRODUCTION		4	21	55
CLUB NEWS	I	PROPOSED REGISTER CONVENTION	M FRANKLIN	5	26	9
CLUB NEWS	I	ACE STANDARD DISK FORMAT		5	28	8
CLUB NEWS	I	QUESTIONS AND ANSWERS ABOUT THE ACE DISK FORMAT	T HILL/W BOWDISH	5	30	16
CLUB NEWS	I	1802 COMPUTER CONFERENCE REPORT		6	31	6
		--				
COMMUNICATIONS	C	RS-232 INTERFACE	B MURPHY	1	5	41
COMMUNICATIONS	C	PACKET RADIO WITH THE 1802	K SMITH/G SIMPDON	2	11	6
COMMUNICATIONS	C	1802 SERIAL I/O	T CRAWFORD	5	25	28
COMMUNICATIONS	C	TELEPHONE DIALING PROGRAM		5	28	20
COMMUNICATIONS	H	UART TO TELETYPE,TELETYPE TO UART, VIDEO/SYNC DRIVER		4	20	31
COMMUNICATIONS	H	A CDP 1854 UART CIRCUIT	E TYSON	5	29	18
COMMUNICATIONS	I	THE EIA RS-232-C STANDARD	ELECTRONIC INDUSTRYS	6	33	13
COMMUNICATIONS	S	ELF II SERIAL INTERFACE	D JORENS	4	21	43
COMMUNICATIONS	S	ELF II SERIAL I/O PACKAGE	W STEINER	4	23	37
COMMUNICATIONS	S	1854 UART OPERATIONS ON THE ACE CPU BOARD	K BEVIS	6	34	36
COMMUNICATIONS	S	9600 BAUD SERIAL I/O FOR 1802	G JONES	7	38	6
		--				
CONTROL	C	INTERUPT PROCESSING ON THE 1802	B MURPHY	1	4	28

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
ERRATA	H	S100 INTERFACE (IF#15)		4	19	3
ERRATA	H	A BETTER BOOT (IF#15)		4	19	4
ERRATA	H	NETRONICS ADAPTER BOARD BUGS		5	28	3
ERRATA	H	ANOTHER BOOT CIRCUIT (IF 21)		5	28	6
ERRATA	H	1802 SERIAL I/O BOARD		5	28	6
ERRATA	H	ACE VDU BOARD		7	37	5
ERRATA	H	A SIMPLE EPROM PROGRAMMER (IF #36 P 9)	D SCHULER	7	37	5
ERRATA	I	ERRATA: IF #14 PAGE 10		4	21	53
ERRATA	I	ERRATA: IF #20 PAGE 35		4	21	53
ERRATA	I	ERRATA: CASSETTE LOAD VOLUME SENSITIVE?		4	21	53
ERRATA	I	ERRATA FOR (IF #20 "MORE BASIC BUGS")		4	22	55
ERRATA	I	MASTERMIND ERRATA (IF #26 P 18)	C GOODSON	6	35	6
ERRATA	S	ERRATA: THE MONITOR - VERSION 2 (IF #20)		4	23	5
ERRATA	S	KALEIDOSCOPE		5	28	6
ERRATA	S	KINGDOM (IF #25 P 18)		5	28	6
ERRATA	S	KINGDOM GAME	C GOODSON	6	35	6
ERRATA	S	ALIEN (IF #35 P 14)	L OWEN	7	37	4
		--				
FORTH	I	FORTH ANYONE	W BOWDISH	4	22	51
FORTH	I	FORTH IS AVAILABLE FOR THE 1802	R COX	4	22	55
FORTH	I	FORTH UPDATE	T HILL	5	28	9
FORTH	I	FORTH IMPLEMENTATION NOTES - ACE SYSTEMS	T HILL	5	29	6
FORTH	I	FORTH - A BRIDGE OVER TROUBLED WATERS	L HART	5	29	20
FORTH	I	ADDING SIMULATED DISK I/O AND A LINE EDITOR TO FORTH	K MANTEI	5	30	37
FORTH	I	FORTH IMPLEMENTATION NOTES - II	T HILL	6	31	8
FORTH	I	FORTH IMPLEMENTATION NOTES - 3	T HILL	6	32	8
FORTH	I	AN EXTREMELY BIASED OPINION OF FORTH	S NIES	6	34	6
FORTH	S	AN 1802 ASSEMBLER FOR 1802 FIG-FORTH	K MANTEI	6	31	10
FORTH	S	1802 FIG-FORTH "MATCH" AND STRING EDITOR	K MANTEI	6	32	10
FORTH	S	MORE FORTH PROGRAMS	T HILL	6	34	8
FORTH	S	USING FIG-FORTH WITH SYSTEMS USING INTERRUPT	T HILL	6	35	7
FORTH	S	AN 1861 TVT FOR FORTH	D RUSKE	6	35	31
FORTH	S	A VLIST FIX FOR FIG-FORTH	R POORE	7	37	15
FORTH	S	AN 1802 ASSEMBLER - FORTH STYLE	S NIES	7	37	19
FORTH	S	FIG-FORTH EXPANSION	T JONES	7	37	31
FORTH	S	FORTH AND THE SMARTERM-80	M SMITH	7	37	35
FORTH	S	FORTH: RIGHT 1802 ASSEMBLY CODE	D HORNER	7	38	25
		--				
GAME	C	HEX PING-PONG	R DELOMBARD	2	9	13
GAME	I	GAME OF LIFE UPDATE - NEW LIFE FORMS	B HUTCHISON JR	2		
GAME	S	LIGHT PATTERN DISPLAY	R GERRISH	1	1	7

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
INTERFACE	C	AN INTERFACE FOR THE MX-80 PRINTER	H SHANKO	5	23	34
INTERFACE	C	1802-APPLE KEYBOARD INTERFACE	J POTTINGER	6	31	19
INTERFACE	C	CENTRONICS PARALLEL PORT FOR THE SUPER ELF	G JONES	6	36	30
INTERFACE	H	A HEXIDECIMAL DISPLAY	F FEAVER	1	3	9
INTERFACE	H	ECONOMICAL HEX DISPLAY	C WILLIAMS	1	4	9
INTERFACE	H	ANOTHER KEYBOARD APPROACH	R KINDIG	1	4	26
INTERFACE	H	RADIO SHACK KEYBOARD ENHANCEMENT	G FOURNIER	2	9	18
INTERFACE	H	ASCII TO HEX CONVERTER	R MACK	2	11	4
INTERFACE	H	A DATA ACQUISITION/CONTROLLER SUBSYSTEM	W GREASON	2	12	4
INTERFACE	H	THE CDP 1853		3	14	30
INTERFACE	H	GETTING A HANDSHAKE FROM THE XITEX VIDEO TERMINAL	K MANTEI	4	20	39
INTERFACE	H	16 KEY KEYBOARD INTERFACE	A TEKATCH	4	20	43
INTERFACE	H	HEX LED DISPLAY	M COYNE	4	24	35
INTERFACE	H	CENTRONICS PRINTER INTERFACE	C VLAUN	4	24	37
INTERFACE	H	1802 TO S-100 BUS CONVERTER	D SCHULER	5	27	18
INTERFACE	H	KEYBOARD BELL CIRCUIT	T CRAWFORD	5	29	11
INTERFACE	H	A PAINLESS ACE-TO-ELFII BUS ADAPTER... THE Q&D	T JONES	5	30	14
INTERFACE	H	A SCANNING HEX KEYBOARD ENCODER	G THOMSON	6	33	33
INTERFACE	I	USING THE 8 DIGIT DISPLAY	W BOWDISH	1	5	49
INTERFACE	I	MICROCOMPUTER INTERFACING, IEEE DEC 77	J DOYLE	2	7	28
INTERFACE	I	S-100 INTERFACE	C KIMTANTAS	3	15	29
INTERFACE	S	A TO D CONVERTER EXPERIENCES	J DAVIS	2	10	4
INTERFACE	S	MORE DEVICE DRIVERS FOR "THE MONITOR"	S NIES	5	26	21
INTERFACE	S	FORTH AND THE SMARTERM-80	M SMITH	7	37	35

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
MATH	S	A SUBROUTINE FOR PSEUDO-RANDOM NUMBER GENERATION	T CRAWFORD	1	1	9
MATH	S	RANDOM NUMBER GENERATOR FOR THE 1802	B MURPHY	1	4	23
MATH	S	A SOFTWARE FOATING POINT MATH PACKAGE	BOWDISH/TOMCZAK/VERL	4	22	5
MATH	S	16-BIT INTEGER ARITHMETIC FOR THE 1802	W BOWDISH	6	32	12
MATH	S	SIMPLE 8 BY 8 BIT MULTIPLY PROGRAM	L KEENLISIDE	7	37	17
MATH	S	SIMPLE 8 BY 8 BIT MULTIPLY PROGRAM	L KEENLISIDE	7	37	17

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
MEMORY	C	EPROM "DISK"	J SWOFFORD	6	34	28
MEMORY	H	MEMORY MAPPED I/O FOR THE 1802	T CRAWFORD	1	4	15
MEMORY	H	ROM OUT OF RAM	M PUPEZA	1	4	25
MEMORY	H	AN 1802 RAM SYSTEM	B MURPHY	1	6	57
MEMORY	H	MEMORY PAGE DECODING	K BEVIS	2	8	35
MEMORY	H	A 16K MEMORY SYSTEM	B MURPHY	2	10	32
MEMORY	H	1802 DYNAMIC RAM CONTROLLER	H SHANKO	3	14	8
MEMORY	H	64K DECODING FOR TEKTRON MEMORY BOARDS	D CARRIGAN	3	16	51
MEMORY	H	TEKTRON 3/4K CMOS RAM MODIFICATIONS	T HILL	4	21	51
MEMORY	H	64K DYNAMIC BOARD	D HELLER/V GOEDHARTN	5	28	14

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
MEMORY	H	1K MEMORY EXPANSION	D RUSKE	5	30	23
MEMORY	H	A MINIMUM COUNT 2114 MEMORY SYSTEM USING THE ACE VDU	F FEAVER	6	31	33
MEMORY	H	THE LAST MEMORY	G JONES	7	37	13
MEMORY	I	MEMORY EXPANSION FOR THE ELF	M FRANKLIN	3	14	35
MEMORY	I	THE MEGABYTE ELF	R SIDDALL	5	27	11
MEMORY	I	MEMORY MAPPING THE 1851	L HART	5	28	37
MEMORY	I	COMMENTS ON "THE MEGABYTE ELF" (IF#27)	J HOWELL	5	30	20
MEMORY	I	USING THE ACE VDU BOARD FOR RAM ONLY	G FEAVER	6	31	33
MEMORY	S	MEMORY TEST ROUTINE FOR THE TEC 1802	A DUNLOP	1	4	20
MEMORY	S	EXTREMELY FAST MEMORY TEST ROUTINE	K SMITH	2	8	30
MEMORY	S	"DOUBTING THOMAS" MEMORY TEST PROGRAM	J CAYER	3	18	51
MEMORY	S	MEMORY TEST PROGRAM	E SMOTHERS	5	25	17
MEMORY	S	NEW MEMORY TEST PROGRAM	E SMOTHERS	5	29	17
MEMORY	S	SHORT MEMORY TEST PROGRAM	L KEENLISIDE	7	38	24
		--				
MISC	C	COMBINATION LOCK AND DOOR CHIME	A TEKATCH	1	6	59
MISC	C	BYTE CONTROL #84	E JACKSON	3	16	43
MISC	C	A CHEAP PRINTER	R THORNTON	3	18	4
MISC	H	MORE POWER TO THE ELF	M FRANKLIN	3	13	17
MISC	H	MEMORY MAPPING THE 1851	RCA	4	22	49
MISC	H	NETRONICS CIRCUITS MODS	J SWOFFORD	5	29	30
MISC	H	A SIMPLE 'CAPS-LOCK' CIRCUIT	W BOWDISH	6	32	21
MISC	H	MINUS 5 VOLTS FOR 64K DYNAMIC BOARD	D STEWART	6	35	40
MISC	I	THE 1802 D VS THE 1802 CD	K SMITH	1	3	3
MISC	I	ABSTRACTS FROM DR. DOBB'S JOURNAL	P BIRKE	1	6	10
MISC	I	1802 PROGRAMMER'S NOTEBOOK	D WRIGHT	1	6	50
MISC	I	1802 OP-CODE TABLE	B MURPHY	2	7	12
MISC	I	HARDWARE BASICS	F FEAVER	2	8	48
MISC	I	MAKE YOUR OWN POWER TRANSFORMERS	K BEVIS	2	11	43
MISC	I	SOME THOUGHTS ON DEVICE INDEPENDANT I/O	W BOWDISH	2	12	30
MISC	I	SOFTWARE PUBLICATION USING BAR CODE	B BLOMMERS	3	14	28
MISC	I	4016'S CAUSE DATA BUS PROBLEMS	K MANTEI	3	17	47
MISC	I	A HARDWARE BUG IN THE 1802	G PICK	3	18	25
MISC	I	TEKTRON NUMBER CRUNCHER	A TEKATCH	4	19	22
MISC	I	THE 1802 AND TTL	R THORNTON	4	19	28
MISC	I	HOME BREW OF PRINTED CIRCUIT BOARDS		4	20	63
MISC	I	DANGER - ELECTRICITY CAN KILL YOU		4	21	6
MISC	I	RECYCLING THE NETRONICS MATH BOARD	M FRANKLIN	4	21	46
MISC	I	CONVERTING FROM 2708'S TO 2716'S	T HILL	4	21	52
MISC	I	A NOTE ON RCA PARTS	T HILL	4	21	52
MISC	I	1802 SUPPORT	L HART	4	22	53
MISC	I	THE EXPANDED 1805 AND 1806 INSTRUCTION SET		5	26	6
MISC	I	ACE SYSTEM CARD CAGE	R POORE	6	32	22
MISC	I	CROSS REFERENCE CHART - 1802 OPCODES TO USR MANUAL PGE	D CLARIDGE	6	33	12

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
PRODUCT	H	MODIFICATIONS TO THE CLUB VDU BOARD	P MUIR	4	23	27
PRODUCT	H	ACE PRODUCT CATALOG		5	29	31
PRODUCT	H	ACE CPU BOARD		6	33	36
PRODUCT	H	ACE PRODUCT CATALOG		7	38	29
PRODUCT	I	NEW PRODUCT ANNOUNCEMENT - ELF II TO ACE ADAPTER BOARD		4	23	4
PRODUCT	I	ACE PRODUCT CATALOGUE		5	25	
PRODUCT	I	ACE PRODUCT CATALOG		5	25	
PRODUCT	I	RCA NEWS	RCA	5	25	5
PRODUCT	I	CTL - THE FAIRLY NEW FULL BASIC COMMAND BY NETRONICS	A MAGNANI	5	25	12
PRODUCT	I	THE NETRONICS SMARTERM-80	M SMITH	6	32	28
PRODUCT	I	ACE FRONT PANEL		6	32	37
PRODUCT	I	1802 COMPUTER PRODUCTS AVAILABLE FROM JOHN WARE	J WARE	7	37	39
PROGRAM	S	LOTTERY TICKET PROGRAM	N/D INKSTER	1	2	14
PROGRAM	S	A NOTABLE ASSEMBLER LOADER	D STEVENS	5	27	21
PROGRAM	S	1802 TINY PILOT	R PETTY	6	33	20
QUEST	H	QUEST DYNAMIC BOARD FIX	L BLOK/C BENTROTSTA	5	28	35
QUEST BASIC	S	QUEST BASIC PRINTER ROUTINE		5	25	23
QUEST BASIC	S	RATRACE 1 - A SIMULATION GAME IN QUEST SUPER BASIC 3.0	P LIESCHESKI	5	25	25
QUEST BASIC	S	SOLVING THE SCHROEDINGER EQUATION WITH THE SUPER ELF	P LIESCHESKI	5	26	27
QUEST BASIC	S	A "CRAPS" PROGRAM FOR QUEST BASIC 3.0	P MUIR	6	33	11
QUEST BASIC	S	INFESTATION II - A SIMULATION GAME IN QUEST BASIC	P LIESCHESKI III	6	33	29
REVIEW	S	NETRONICS FULL BASIC REVIEW	M FRANKLIN	3	16	19
REVIEW	S	MORE BASIC BUGS	T POLLARD	3	18	29
TINY BASIC	C	VIP SOFTWARE UPDATE	R BLESSING	2	8	7
TINY BASIC	H	A LOW COST 4K TINY BASIC	G GILBERT	3	17	38
TINY BASIC	I	NETRONICS TINY BASIC CAN RUN ON ALL 1802 COMPUTERS	H STUURMAN	4	23	33
TINY BASIC	I	IMPLEMENTING QUEST TINY BOARD	W SWINDELLS	5	29	15
TINY BASIC	I	NOTES ON NETRONICS TINY BASIC	O HOHEISEL	6	33	35
TINY BASIC	S	TINY BASIC ON THE VIP SYSTEM	R BLESSING	2	7	16
TINY BASIC	S	A TINY BASIC SQUARE ROOT ROUTINE	T CRAWFORD	2	7	44
TINY BASIC	S	BLACKJACK IN TINY BASIC	J SMITH	3	14	18

KEYWORD	CLASSIFICATION	TITLE	AUTHOR	YEAR	ISSUE	PAGE
VIDEO	S	EDITOR/ASSEMBLER/MONITOR FOR VIDEO	T JONES	4	21	41
VIDEO	S	SOFTWARE FOR THE ACE VDU BOARD	T HILL	6	31	16
VIDEO	S	AN 1861 ADDRESS/DATA EXAMINE/LOAD FUNCTION	J CAYER	6	34	30
VIDEO	S	ADDING SCRT TO THE WINDOW PROGRAM	T HILL	6	35	11
VIDEO	S	CHIP - 8 FOR THE ACE VDU BOARD	T HILL	6	35	12
