Ipso Facto

ISSUE 35 JUNE, 1983

INDEX PAGE

A PUBLICATION OF THE ASSOCIATION OF THE COMPUTER-CHIP EXPERIMENTERS (ACE) 1981

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.

March 2, 1983

Dear Mike,

I was very concerned about the club and its future after reading the letters from Wes Steiner and Fred Hannon, and your response. Personally, computer hardware is much more interesting to me than the software, and I hope the hardware orientation will continue. For software, the Questdata newsletter is nice, and there are many magazines devoted to programs and programming in BASIC. IPSO FACTO is the only publication dedicated to hardware at the hobbyist level so far as I know.

Your comments concerning article submissions encouraged me to go through back issues for more information. I found that only the 68 members below have contributed articles in the last 2.5 years (since issue 19). The table below shows these authors, their locations (C=ACE Headquarters area, A=American, O=other), and 5 columns of numbers showing the number of articles printed in issues 19-21, 22-24, 25-27, 28-30, and 31-33. My apologies for any errors or omissions.

W Bowdish	C	12015	L Blok	O	00010	C Bouwhuis	O	00010
D Bauer	A	00200	K Bevis	C	02001	G Bertrand	O	01000
J Cayer	O	10100	T Crawford	C	00110	G Caughman	A	00100
R Cox	A	11000	M Coyne	C	01000	D Doerr	C	10000
B Eckel	A	20000	B Erskine	A	00010	M Franklin	C	10342
F Feaver	C	00003	R Francis	C	01000	O Hoheisel	O	00001
J Howell	A	00010	F Hannan	A	00001	J Hart	A	01120
T Hill	C	10036	D Heller	O	00010	H Hallaska	A	00010
A Irwin	A	00001	D Jorens	A	11000	T Jones	A	10230
T Jones	O	00001	P Liescheski	A	00301	E Leslie	C	10000
A Magnani	A	00100	K Mantei	A	52012	J McDaniel	A	00100
P Muir	C	02001	J Munck	A	00010	J Munch	A	00001
S Nies	A	11110	T Pittman	A	10101	A Pacheco	A	01000
K Poore	A	00001	J Pottinger	A	00001	D Ruske	A	01110
V Raab	O	10000	D Stevens	A	00100	W Steiner	A	02001
H Shanko	A	22010	J Swofford	A	00110	D Schuler	A	21110
K Schultz	A	00010	D Schroyer	?	10000	W Swindells	C	00010
J Stephens	A	00100	E Smothers	A	00120	H Stuurman	O	11000
B Smith	C	00001	M Smith	C	00001	E Shaffer	A	01000
R Siddall	C	00220	T Setaro	A	00010	A Tekatch	C	30000
D Taylor	A	10000	R Thornton	A	10010	G Tomczak	C	01000
E Tyson	A	00020	G Thomson	O	00001	J Vaal	A	00100
R Verlaan	?	01000	C Vlaun	O	01000			

A quick glance down the rightmost column shows that the last 3 issues were written almost exclusively by Wayne Bowdish, Mike Franklin, Fred Feaver, and Tony Hill. While these people are fine authors and very knowledgeable, they are soon going to tire of entertaining the remaining 496 or so of us, and will quit in disgust. I made some graphs of the data in the chart, and found that American members contributed 52%-73% of the articles printed between issue 19 and issue 27. This dropped sharply from that time until the present, when only 30% were contributed by Americans in the last three issues. Our A.C.E. Headquarters people have picked up the load, and saved the newsletter. Article submissions from the "other" group have been remarkably constant over the period shown. While the decline in American submissions may be due to apathy or lack of interest, I believe it is due to economic problems caused by the recession in America, which has many people out of work. If so, I hope the trend will reverse soon.

You made several important points in your answer to Fred Hannan: (1) you print what you receive from members, (2) members who write articles do so at their current level and about their current work, and (3) few members care enough about the Club to contribute to the newsletter.

With so few people contributing, this is no time to complain about what is being submitted. I feel it would be far better to submit an article of some kind concerning your own interests and hope to stimulate others of similar inclination to follow suit. Everyone who owns a computer has something to share if he will think about it. If you cannot think of anything at all to write, look through back issues for an interesting article and write an encouraging letter to its author for more of the same. Letters to the Editor are an interesting part of the newsletter, as well.

Three cheers for you few who are keeping the newsletter alive. Hang in there a little longer and maybe we'll all submit an article or two.

Dick Thornton

Dick Thornton
1403 Mormac Road
Richmond, Va. 23229

Editor's Comment: Thanks, Dick, for the comments, the interesting statistics and the articles you submitted. As a point of interest, the current recession has hit Canadians and our Overseas members too. If Canadians sit down and use their computers and write articles when unemployed, what do Americans do? I would hope that they do the same thing. To repeat what I said in the last Newsletter - learning micro-computer technology and its applications could save your job, or help you get a new one. M.F.

Kendall Stambaugh, Ph.D.
5009 Guide Meridian, Bellingham, Wa. 98226

Dear Mike:

I just wrote to you, but that was before I received IF 33. I echo your last line to Fred; if it happens, I too shall be saddened by its passing. I hope it doesn't.

I am somewhere between Fred's Low tech, which wants to stay with the ELF, and the High Tech that wants to junk the ELF and replace EVERYTHING with newly designed boards. I just want to upgrade my ELF.

One thing Fred Hannan said really struck a nerve. "Those members who are fortunate enough to be able to attend the meetings in person ....",etc. Maybe it is inevitable that almost everything in ACE is for the benefit of these members, but occasionally you folks should stop to remember that <u>we</u> pay the same dues as they do.

Enough train of thought, now. Although I dump on you, I appreciate what you are doing for IF, and I hope you are willing to continue.

Editors Comment:

The local meetings seem to be somewhat of a sore point among our distant members. Just remember: the local folk keep the Club functioning, without special compensation, so you have a newsletter to read. Club meetings don't cost anything to have in dollars - just time and personal effort, but it keeps people interested and doing things, and writing articles, and putting together newsletters, and mailing boards and newsletters, and running around, and designing new boards, and organizing conferences so <u>you</u> have something to belong to, and to enjoy, and to benefit from as well. MF

Dear Mike:

Four years ago I responded to a letter very similar to Fred Hannan's in I.F. 33. My letter was published in I.F. 11, page 50. My opinion hasn't changed.

Much to my displeasure, my 1802 hasn't changed either. The main reason for this is that I just didn't know what to do about it. Due to its "low tech" personality, I've found myself spending very little time with it.

Contrary to Fred's opinion that, "asking the average member to replace his motherboard is just driving him away", in my case, at least, the Club is providing a means of keeping me.

I don't want to buy a personal computer; I want to build one and know it inside out. The "high tech" members of ACE, through their hard work and willingness to share, are providing me with the material that I need to do just that.

Talk of the Club folding at a time when I thought its maturity looked serious enough to make a commitment to, is disheartening, to say the least.

As for "where does that leave us who cannot attend" meetings - isn't that what the Newsletter is for? I don't recall seeing any articles describing a problem that couldn't be resolved and needed some help. Are we "low tech" people too shy to ask for help?

I, for one, will not be. I plan to "high tech" my 1802. I will document my progress in IPSO FACTO, and request assistance when needed.

Mike, I hope that you get swamped with similar letters in order that you and the other "workers" be encouraged to continue to do the fine job you have been doing.

Yours truly,

Dave Robinson
6528 Montrose Trail, Talahassee, Fl. 32308

Editors Comment:
Dave, thank you for writing and for your board order. I don't think ACE will fold - there are enough local members to keep it going as a <u>meeting</u> club. But, it will only continue as a <u>publishing</u> club if our members continue to submit articles, and right now, their interest appears to be picking up satisfactorily. M.F.

Bugs

- by C.C. Goodson, Campinas, S.P. Brazil

There appears to be a bug in the listing of the MASTERMIND modifications program of IF 26 p. 18: Addresses 86 to C0 are indicated as not modified from the original program, but in the dump list included, for those who do not have IF 10, address B8 is listed as B5 when it should be BD. Also address 85 is listed in the dump at 17, but as DP in the modifications list. The latter is correct.

The TBasic KINGDOM Game (IF 25 p.18)) has an error of omission at the end of line 500, where the math operator is omitted between the last 2 items: 19F. Should this be 19+F, 19-F, or 19*F? Except for this doubt, the game runs fine, leaving one quickly frustrated as he is deposed.

The Decimal to Hex Conversion Routine of Mr. Caughman's Tiny Basic Programs (IF 27 p.8) has a misprint on line 570 of page 9: The quotation marks should be deleted, so as to read 570 PRINT V; On page 10, line 570 is correct.

USING fig-FORTH WITH SYSTEMS USING INTERRUPTS

by: Tony Hill 30-481 Pitfield Rd. Milton Ontario

A number of club member have run into trouble when attempting to use FORTH with an 1861 output device. The problem stems from the fact that the 1861 is interrupt driven. The author of 1802 FORTH (or more correctly - the authors) did not always stick to a correct method of using the R2 stack. They allowed the stack pointer to advance back past data that they intended to use later. This tends to produce funny results if an interrupt comes along and the interrupt routine also tries to use the R2 stack. The problem occurs if the FORTH word I is used to fetch the loop index onto the data stack.

Fortunately, there is an easy fix for the problem. It involves modification to the interrupt routine so that it decrements R2 twice BEFORE it pushes anything onto the stack. If it is not possible to modify your interrupt routine, you could try patching the FORTH I word. It is located at 1425 in the club version, and a few bytes up from that in the official fig version. The code looks like this

```
1425  12    INC  R2
1426  19    INC  R9
1427  19    INC  R9
1428  19    INC  R9
1429  42    LDA  R2     ; R2 now 2 bytes up !!
142A  59    STR  R9
142B  29    DEC  R9
142C  02    LDN  R2
142D  59    STR  R9
142E  22    DEC  R2
142F  22    DEC  R2
1430  DC    SEP  RC
```

If an interrupt occurs between 1429 and 142F the R2 stack will be clobbered. FORTH uses the R2 stack to nest return addresses so clobbering its contents will tend to send it off into never-never land. I can't think of a way to rewrite this code in the smae space to fix the problem, so it is up to the user to fit a patch in where ever he can find room in his system.

AN 1802 THREADED CODE IMPLEMENTATION

by: Ed Redman RR #2 Porters Lake NS B0J 2S0

INTRODUCTION TO THREADED CODE

Threaded code is a term used to indicate that a program consists simply of a set of links to other programs. The other programs could be machine code subroutines or other sets of links.

One means of producing threaded code is to program a whole set of subroutine calls (i.e. SCRT calls) like this:

```
0200  D4 0304   CALL 0304
0202  D4 0206   CALL 0206
0204  D4 0609   CALL 0609
```

Such a program would be unnecessarily large, as every third byte would be a D4. If we remove all the D4's we could interpret the remaining addresses with a small interpreter program. In FORTH this interpreter is called NEXT.

The FORTH word NEXT is different from the implementation I use. The interpreter I use requires and is used in conjunction with RCA's standard call and return technique - SCRT. FIG-FORTH does not use SCRT or the reserved SCRT registers.

Use of my method requires :

1. R2 to R5 setup as per SCRT
2. R6 points to the first byte of threaded code
3. One other register for the interpreter (I use RC)

INT (Interpreter)

My interpreter, called INT, allows me to use threaded code directly and is quite fast when compared to the regular CALL and RETURN. INT requires as few as 6 bytes per CALL and RETURN as opposed to as many as 33 for SCRT. INT may be made slightly longer for ease of use; the optional instructions are shown in brackets.

```
                SEP R3         ; EXIT POINT TO MACHINE CODE
RC-->     INT: (PHI RF)        ; ENTRY - SAVE RF (OPTIONAL)
               (SEX R2)        ; R2 = STACK      (OPTIONAL)
                LDA R6         ; GET NEXT ADDRESS
                PHI R3         ;  AND PUT IN R3
                LDA R6         ;
                PLO R3         ;
               (GHI RF)        ;
                BR  INT-1      ; LOOP BACK AND EXIT
```

NOTES:

1. The stack is not disturbed.
2. R6 is available to pass data or point to the next threaded code byte.
3. The called subroutine must be in machine language.
4. The exit from the subroutine is a SEP RC .

This routine is all that is needed to execute threaded code. The rest of this article deals with making it easier to use.

NESTING THREADED CODE ROUTINES
By itself, INT allows only machine code routines to be called. Nested routines written in threaded code require the use of a routine called T-CALL. T-CALL allows threaded code routines to call other threaded code routines if a CALL T-CALL is placed at the start of the routine being called.

Returning to the calling program requires (you guessed it) a T-RET routine. Since a CALL T-CALL places R6 on the stack, T-RET must simply restore R6 and execute a SEP RC.

The code for T-CALL and T-RET is shown below:

```
        0229 DC      SEP  RC ; T-CALL
        ....
        0253 12      INC R2  ; T-RET
        0254 42      LDA R2  ;
        0255 B6      PHI R6  ;
        0257 02      LDN R2  ;
        0258 A6      PLO R6  ;
        0259 9F     (GHI RF) ;
        025A DC      SEP RC  ;
```

An example of the use of T-CALL and T-RET is

```
        ....
R6-->   047E 0604                    ; THREADED CODE CALL TO 0604
        ....
        0604 D4 0229  CALL T-CALL ; CALL TO NEST ROUTINE
        0607 0709                 ; MORE THREADED CODE ADDRESSES
        0608 0342                 ;
        060A 0253                 ; CALL TO T-RET  TO UNNEST
        ....
```

NOTE: Check your version of SCRT for order R6 is placed on stack.

MIXING MACHINE CODE AND THREADED CODE
Calling threaded code routines from machine code is quite easy. To begin a threaded code sequence just CALL T-CALL. To return to machine code requires an UNTHREAD routine, which is just a SEP R5.

EXAMPLE:

```
         ....
         0229   DC       SEP   RC      ; T-CALL
         ....
         0275   D5       SEP   R5      ; UNTHREAD ROUTINE
         ....
R3-->    0700   F3       XOR           ; SAMPLE CODE
         0701   B9       PHI   R9      ;
         0702   D4 0229  CALL T-CALL   ; LINK IN THREADED CODE
         0705   0203     SUB1          ; THREADED ADDRESSES
         0707   0689     SUB2          ;
         0709   0275     UNTHREAD      ; RETURN TO MACHINE CODE
         070A   FF FF    SMI   FF      ;  AND CONTINUE
         ....
```

BRANCHING AND SKIPPING
Branching about in threaded code requires a few routines. For speed short branches can be used. For convience long ones are handy. (NOTE: no machine code LBR's are necessary)

Short Branch:

```
        ....
        0942 46         LDA R6    ; GET INLINE ADDRESS BYTE
        0943 A6         PLO R6    ; PUT INTO THREADED CODE POINTER
        0944 DC         SEP RC    ; RETURN TO INTERPRETER
Long Branch:
        0945 46         LDA R6    ; GET INLINE HIGH BYTE
        0947 52         STR R2    ; SAVE IT
        0948 46         LDA R6    ; GET INLINE LOW BYTE
        0949 A6         PLO R6    ; PUT INTO THREADED CODE POINTER
        094A 02         LDN R2    ; RESTORE HIGH BYTE
        094B B6         PHI R6    ; PUT INTO THREADED CODE POINTER
        094C DC         SEP RC    ; RETURN TO INTERPRETER
Skip:
        094D 16         INC R6   ; INCREMENT THREADED CODE POINTER
        094E 16         INC R6   ; ...PAST NEXT LINK
        094F DC         SEP RC   ; RETURN TO INTERPRETER
        ....
Example use:
        ....
        0B21 0942 28             ; SHORT BRANCH TO 0B28
        0B24 0945 8000           ; LONG JUMP TO MONITOR AT 8000
        0B28 094d                ; SKIP NEXT LINK
        0B2A 0275                ; SKIP JUMPS THIS LINK
        0B2C 0700                ; SKIP COMES HERE
        ....
```

CONDITIONAL BRANCHING EXAMPLE

Short branch if DF = 1

```
        ....
        0982 33 42      BDF SHORT ; DO BRANCH CODE IF DF=1
        0984 26         INC R6    ; OTHERWISE SKIP INLINE BYTE
        0985 DC         SEP RC    ; RETURN TO INTERPRETER
Long branch if DF = 1
        0986 33 45      BDF LONG  ; DO BRANCH CODE IF DF=1
        0988 30 4D      BR  SKIP  : OTHERWISE SKIP INLINE WORD
        ....
```

I have been experimenting with threaded code for over a year. It is fast and compact. For those 1861 I/O users (DOTS etc.) I have a video program which displays 16 lines by 16 characters. I will gladly send a copy (HEX dump) to any who request it. It requires 2K (1 for display).

ADDING SCRT TO THE WINDOW PROGRAM

by- Tony Hill 30-481 Pitfield Rd. Milton Ontario

Back in the July 82 issue of Ipso Facto I published my version of an 1802 debugging tool which I called WINDOW. This program provided a full screen emulation of an 1802, showing what was going on in its registers and what instructions it was executing.

I received many letters about the program, and for the benefit of those who did not write, the code listed in Ipso was correct as printed. However, I have come up with one small improvement.

The original version of WINDOW emulates code exactly as it is found in memory. While this is fine for most debugging, it creates a problem if the program contains a lot of SCRT calls. It quickly becomes very tiring watching the 31 SCRT instructions flash by everytime a subroutine is called. You really get some idea of the overhead involved in using SCRT, and debugging becomes tedious.

Listed below is a patch to mask out the SCRT instructions in the TRACE and STEP modes. The patch is inserted in some unused memory on pages 3 and 4 of WINDOW. The patch will be used if you change the byte at 0311 from 7F to F3.

The SCRT patch works by switching WINDOW into QUICK mode whenever it sees a D4 or a D5 instruction. It stays in QUICK mode until it finds another Dx type instruction (presumably a D3). If the program being tested has R4 and R5 pointing to valid SCRT code that code will be executed without being displayed.

ADDRESS	DATA	ADDRESS	DATA
03F3	D7 8B 8C F6	04F2	D4 02 4F 9C
03F7	FF 02 3A FE	04F6	FF 0D 32 FF
03FB	D4 04 F2 D5	04FA	D4 01 CA 3A
		04FE	F2 D5

There is one other note I should throw in here. If WINDOW comes to a branch on EFx instruction, it asks for the value of that flag line. If you do not enter ANY value, but just press a carriage return WINDOW will attempt to execute the branch address that follows as an instruction!! This can cause funny things to happen, so don't do it. However, if you insist on being fumble fingered, and don't plan to use the patch listed above, the extra memory can be used for the following patch-

ADDRESS	DATA	ADDRESS	DATA
038A	D4 03 F3 C4	03F3	D4 00 90 F3
		03F7	D5

Work on my serial version of WINDOW has ground to a halt. I have it coded and mostly tested, but haven't been able to find time to finish. Such is life.....

CHIP - 8 for the ACE VDU Board

by: Tony Hill 30-481 Pitfield Rd. Milton Ontario

Many years ago when the Radio Corporation of America was still interested in promoting their microprocessor chip as something other than a laboratory curiosity, they developed a simple home computer built around it. This computer, known in the dark old days as the VIP had a simple interpreter that could be entered into it to run games and other equally useless programs. The only interesting thing about the interpreter (CHIP-8) is that many games were published for it by RCA and others.

This article has a HEX listing of a version of that interpreter that can be run on any system using the ACE VDU board. This makes the library of CHIP-8 games available to systems using that board. The program is loaded at address 1000 and requires three pages. The ACE VDU board will require one simple mod to make the program run.

The original CHIP-8 ran with an 1861, which provided an interrupt every 1/60 seconds, usefull for timing purposes. There is a pin on the VDU 6847 chip that can also provide that signal, but it must be tied to a flag line and polled in software instead of generating interrupts. The listing assumes EF1 will be used, but it is possible to use any of the four lines by changing the underlined 34 and 3C branch instructions accordingly. Simply solder a wire between pin 37 (FS) and edge connector pin 19.

Input to CHIP-8 was originally done with a scanned HEX keypad. Since it is not likely that an ACE system would have the same hardware configuration, two long branch instructions have been provided to allow the user to patch in his own input. The branch at 110D should jump to a routine that gets a single HEX digit and puts it in D. The routine should set DF and return with a SEP R4 (D4) instruction. The routine jumped to at 119C should check for a key pressed, and set DF if so or clear it if not. If pressed, the key value is placed in D. Either way the routine should return immediately with a SEP R4. Note that CHIP-8 uses the SCRT registers internally, but not for SCRT. Registers C,D,E,F are available for I/O use, all others must be saved first. R3 is the program counter and R2 points to the first free byte of a grow down stack.

The program is used in the same manner as the original CHIP-8. CHIP-8 programs are loaded at 0200, and program execution starts at 1000 with R0 or R3 as the program counter. One other interestin note is that CHIP-VDU has four times the video resolution of the original program, thanks to the use of the 6847 instead of the 1861.

One final note- the address of the control register for the 6847 video modes is stored at 11EC for the high byte and at 11EF for the low byte. The last used address of the program is 12EE.

1000	F8	0F	B2	B6	F8	CE	A2	F8	10	B4	F8	1E	A4	F8	11	B5
1010	F8	FC	A5	F8	CF	A6	F8	00	B8	A8	56	C0	11	EB	F8	CF
1020	A6	06	34	2B	32	40	F8	00	56	30	40	3A	40	F8	01	56
1030	98	32	36	FF	01	B8	88	32	3F	28	31	3E	7B	38	7A	19
1040	96	B7	E2	94	BC	45	AF	F6	F6	F6	F6	32	87	FC	68	AC
1050	8F	FA	0F	F9	F0	A6	05	F6	F6	F6	F6	F9	F0	A7	4C	B3
1060	8C	FC	0E	AC	0C	A3	D3	30	1E	11	11	11	11	11	11	11
1070	11	11	11	11	11	12	11	11	7C	75	83	8B	95	AD	B0	B5
1080	91	E4	9F	D2	43	99	05	8F	32	90	FA	0F	B3	45	30	65
1090	05	FF	E0	32	9C	FF	0E	32	CE	8F	30	8A	3C	B6	F8	CF
10A0	A6	06	3A	B4	19	98	32	AB	FF	01	B8	88	32	B4	28	31
10B0	B3	7B	38	7A	34	B4	3C	B6	F8	FF	AF	F8	E3	BF	EF	F8
10C0	00	73	8F	3A	BF	9F	FF	E0	3A	BF	73	15	30	2D	42	B5
10D0	42	A5	30	1E	56	33	1E	F8	0A	A3	F8	11	B3	D3	E6	F3
10E0	FA	0F	A3	45	76	33	F1	FE	3B	1E	83	3A	1E	15	15	30
10F0	1E	FE	3B	ED	83	3A	ED	30	1E	FF	FF	FF	FF	FF	FF	FF

1100	FF	FF	FF	FF	00	45	A3	98	56	D4	F8	D4	A4	C0		
1110	FF	FF	FF	FF	00	06	B8	D4	06	A8	D4	64	0A	01	E6	8A
1120	F4	AA	3B	28	9A	FC	01	BA	D4	F8	12	BA	06	FA	0F	AA
1130	0A	AA	D4	E6	06	BF	93	BE	F8	1B	AE	2A	1A	F8	00	5A
1140	0E	F5	3B	4B	56	0A	FC	01	5A	30	40	4E	F6	3B	3C	9F
1150	56	2A	2A	D4	00	22	86	52	F8	F0	A7	07	5A	87	F3	17
1160	1A	3A	5B	12	D4	22	86	52	F8	F0	A7	0A	57	87	F3	17
1170	1A	3A	6B	12	D4	15	85	22	73	95	52	25	45	A5	86	FA
1180	0F	B5	D4	45	E6	F3	3A	82	15	15	D4	45	E6	F3	3A	88
1190	D4	45	07	30	8C	45	07	30	84	F8	DE	A4	C0			F8
11A0	F0	A7	E7	45	F4	A5	86	FA	0F	7C	00	B5	D4	45	56	D4
11B0	45	E6	F4	56	D4	45	FA	0F	3A	BD	07	56	D4	AF	22	F8
11C0	D3	73	8F	F9	F0	52	E6	07	D2	56	F8	FF	A6	F8	00	7E
11D0	56	D4	19	89	AE	93	BE	99	EE	F4	56	76	E6	F4	B9	56
11E0	45	F2	56	D4	45	AA	86	FA	0F	BA	D4	F8	FF	BF	F8	FF
11F0	AF	F8	32	5F	D4	FF	FF	FF	FF	FF	FF	00	00	E0	12	00

1200	30	39	22	2A	3E	20	24	34	26	28	2E	18	14	1C	10	12
1210	F0	80	F0	80	F0	80	80	80	F0	50	70	50	F0	50	50	50
1220	F0	80	F0	10	F0	80	F0	90	F0	90	F0	10	F0	10	F0	90
1230	F0	90	90	90	F0	10	10	10	10	60	20	20	20	70	A0	A0
1240	F0	20	20	06	FA	07	BE	06	FA	7F	F6	F6	F6	22	52	F8
1250	38	BC	07	FA	3F	FE	FE	FE	AC	9C	7E	BC	8C	FE	F1	AC
1260	9C	7E	BC	45	FA	0F	AD	A7	F8	D0	A6	F8	00	AF	87	32
1270	8A	27	4A	BD	9E	AE	8E	32	82	9D	F6	BD	8F	76	AF	2E
1280	30	76	9D	56	16	8F	56	16	30	6B	8D	A7	87	32	93	2A
1290	27	30	8C	EC	F8	00	A7	3C	B1	F8	CF	A6	06	3A	AF	19
12A0	98	32	A6	FF	01	B8	88	32	AF	28	31	AE	7B	38	7A	34
12B0	AF	3C	B1	F8	D0	A6	8D	32	E2	06	F2	2D	32	C1	F8	01
12C0	A7	46	F3	5C	02	FB	0F	32	D5	1C	06	F2	32	D1	F8	01
12D0	A7	06	F3	5C	2C	16	8C	FC	10	AC	9C	7C	00	BC	FF	E4
12E0	3B	B6	F8	FF	A6	87	56	12	F8	CF	A6	F8	2D	A4	D4	

Alien - A Game for the 1861

- by Larry Owen, 21A Regina Road, Trenton, Ontario. M8V 1G6

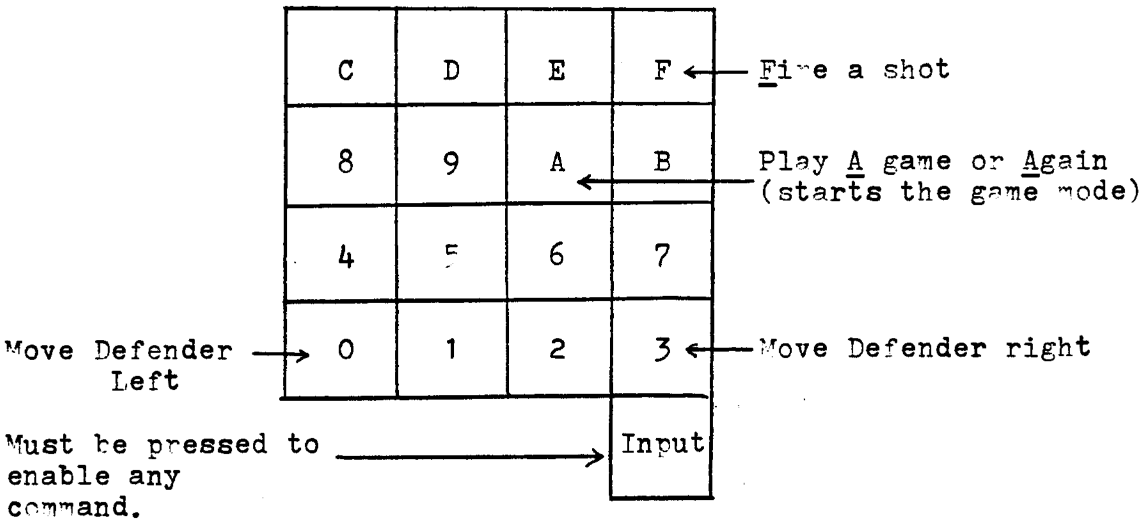
This game is modelled after the Arcade style games. When it first comes up, it is in the "attract" mode, where it alternates between showing the top five scores, and giving a (rather poor) demonstration of how the game is played.

In the game playing mode, there are two types of aliens which rain down towards the defender at the bottom. Shooting the small ones earns a score of onepoint, while shooting the large ones earns five points. The small ones, although appearing at random, come down in rather copious quantities. They are relatively harmless, as they can only take one of your three lives by landing on top of you. But don't get too close to them when they strike the bottom of the display, as they tend to splat a bit, and can wipe you out that way! The large ones appear less often, but are more dangerous; they don't have to hit you to claim one of your lives, they only have to make it to the bottom. As the defender, you are able to move back and forth across the bottom of the display, and to shoot at your tormenters. Your two spare lives are shown at the top center of the screen. When you lose a life, one of the spares disappears from the top, and reappears at the bottom, always in the center. The game is over when you have lost all your lives, or when all 200 of the little beasties have rained down. The number of little beasties left to come is shown in the top left corner of the screen, while your score is shown in the top right corner. There are two objects to this game. One is just to survive to the end. The second is to rack up the highest score you can. I can break 200 fairly regularly, but only once have I managed to wrap the score past the maximum displayable of 255.

This program requires 2K of RAM starting at address 0000. The I/O assignments are:

002C	61	Turn on 1861 video
00AF	3F	Branch if INPUT not pressed
00B8	3F	"
01E4	3F	"
01E9	37	Branch if INPUT is pressed
0230	3C	1861 Status check
0239	3C	"
0240	34	"
01E6	6C	Input from Hex Keyboard
01E7	64	Output to Hex Display
00B2	6C	Input from Hex Keyboard
00BB	6C	"

The following diagram shows the layout of my keyboard, along with what key causes what action:



For other keyboard layouts, you may wish to change these commands. They are located as follows:

00 Left	03 Right	0A Again	0F Fire
0425	042B	0437	0431
065A	065C		0660
065B	065E		0667
065D	065F		
0662	0661		
0663	0665		
0664	0668		
0666	0669		

This program is actually two programs. The first, residing at 0000 to 02FF, is a hexadecimal interpreter, very much like RCA's CHIP 8. I wrote this program after coming across an article in BYTE Magazine which described the CHIP 8 instruction set and how to use it. My version is slightly expanded, and although I believe other CHIP 8 programs could be rewritten to run on my version, I don't think it would be quite so easy to go the other way. If there is enough interrest, I could write a future artical on my version giving the instruction set, how to use it, and a detailed listing (if the editor can spare the pages).

The game program, in interpretor code, resides at 0300 to 06FF. The video refresh RAM is 0700 to 07FF. Happy playing, and watch out for those big beasties! By the way, a speaker appropriatly connected to Q will provide sound effects for the game.

"Alien" - A Game For the 1861

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	C0	00	03	90	B4	B7	A5	A6	F8	0C	A4	D4	F8	02	B1	B2
0010	B8	BA	BB	F8	03	B5	F8	17	A1	F8	EC	A2	F8	7B	A8	EA
0020	F8	EF	AA	F8	03	A7	48	73	27	87	3A	26	61	F8	ED	AB
0030	4B	BE	F8	FF	AE	4B	B9	94	A9	94	59	19	2E	9E	3A	39
0040	38	D3	05	3A	48	15	45	A4	F9	F0	AA	FA	0F	BF	45	FA
0050	F0	F6	F6	F6	FC	4F	A8	05	F6	F6	F6	F6	F9	F0	AB	EA
0060	97	32	67	31	67	7B	38	7A	48	B3	48	A3	30	41	F8	7E
0070	30	1E	9C	BD	8C	AD	30	42	12	42	A5	02	B5	30	42	9C
0080	52	9D	BC	02	BD	8C	52	8D	AC	02	AD	30	42	E2	9C	73
0090	8C	73	30	42	12	42	AC	02	BC	30	42	45	FA	03	FC	6F
00A0	A8	08	A3	0B	F3	3A	A9	15	15	D4	0B	F3	3A	A7	D4	3F
00B0	A9	E2	6C	EA	FA	0F	30	A4	3F	A7	E2	6C	EA	FA	0F	30
00C0	AB	45	30	A4	45	30	AB	45	F4	5A	F8	FF	AA	94	7E	5A
00D0	D4	45	FA	07	FC	73	A8	08	A3	0B	5A	D4	0B	F2	5A	D4
00E0	94	AE	0A	32	EC	EB	38	1E	F7	33	E7	F4	52	8E	5A	F8
00F0	FF	AA	02	5A	D4	0B	30	C8	0B	F5	30	C9	45	A5	9F	B5
0100	D4	96	5A	45	F2	5A	D4	F8	F0	AA	05	F4	A5	9F	7C	00
0110	B5	D4	FF	00	30	18	FC	00	45	FA	0F	32	11	A7	E2	9C
0120	73	8C	73	33	20	9D	BC	8D	AC	0A	FA	07	BF	0A	FA	3F
0130	F6	F6	F6	52	F8	EE	A8	0B	FE	FE	FE	F1	A9	F8	FF	AB
0140	94	5B	08	7C	00	B9	FD	07	33	4C	08	B9	9F	AF	94	AE
0150	4C	BE	8F	32	5E	2F	9E	F6	BE	8E	76	AE	30	52	9E	52
0160	09	F3	59	F3	F2	73	19	8E	52	09	F3	59	29	F3	F2	12
0170	F1	52	0B	F1	5B	27	87	32	80	89	FC	08	A9	99	30	43
0180	12	42	AC	02	BC	D4	45	A3	8C	F4	AC	D4	0A	FA	0F	FC
0190	81	A8	08	AD	98	BD	D4	0A	F6	F6	F6	F6	30	8F	94	AE
01A0	0A	38	1E	FF	64	33	A2	FC	64	52	8E	5C	94	AE	02	38
01B0	1E	FF	0A	33	B0	FC	0A	1C	1C	EC	73	8E	73	D4	8A	FA
01C0	0F	A7	F8	F0	AA	38	27	72	5C	1C	87	3A	C6	D4	8A	FA
01D0	0F	A7	F8	F0	AA	38	27	4C	5A	1A	87	3A	D6	D4	86	73
01E0	D4	0A	A6	D4	3F	E4	6C	64	2A	37	E9	D4	0A	B7	D4	E2
01F0	45	AF	95	73	85	73	9F	B5	8F	A5	D4	45	AD	9F	BD	D4

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0200	76	52	96	FC	01	B6	86	32	0A	26	97	32	10	FF	01	B7
0210	72	7E	72	AB	F0	78	70	C4	22	73	8B	73	F8	EE	AB	4B
0220	B0	94	A0	4B	A1	E2	80	E2	20	A0	E2	20	A0	E2	20	A0
0230	3C	25	30	00	80	E2	20	A0	E2	3C	34	80	E2	20	A0	E2
0240	34	3B	30	00	45	AF	DF	45	AC	9F	BC	D4	45	5A	D4	02
0250	43	00	FC	01	EF	00	C1	00	C4	00	9B	02	4C	00	C7	00
0260	D1	01	FB	02	47	01	07	01	01	01	12	01	16	01	86	A3
0270	AA	AF	B8	D9	E0	DC	F5	F8	F4	F4	F4	26	07	01	34	06
0280	02	91	9A	A5	A3	93	C5	A7	95	A9	AB	9E	C0	AF	BC	B3
0290	B7	E0	A0	A0	A0	E0	20	20	20	20	40	40	40	40	40	A0
02A0	E0	A0	A0	E0	20	E0	20	E0	80	E0	A0	E0	A0	E0	20	E0
02B0	80	80	80	E0	80	C0	80	E0	80	C0	80	80	C0	A0	A0	A0
02C0	C0	A0	C0	A0	C0	E0	80	E0	20	E0	00	00	00	00	00	00
02D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
02E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
02F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0300	16	E0	61	31	60	00	A6	AE	F0	BE	71	FF	31	00	13	08
0310	6E	00	6D	C8	23	8C	23	9C	6C	03	A6	96	96	AB	F1	CE
0320	E0	13	7C	FF	3C	00	13	1E	A6	AF	F1	BE	00	78	00	8D
0330	A6	D2	F7	BE	00	94	00	78	23	2E	00	8D	A6	DA	65	00
0340	64	00	63	00	75	01	F0	CE	00	8D	A6	CF	F0	9E	F2	CE
0350	F5	8C	E3	45	73	10	23	6E	74	06	00	94	35	05	13	42
0360	00	94	00	8D	A6	D2	F7	CE	00	94	00	78	64	00	30	00
0370	13	7A	31	00	13	7E	73	0A	13	86	F0	8C	E3	45	73	05
0380	F1	8C	E3	45	73	05	F2	8C	E3	45	00	78	23	2E	00	8D
0390	A6	CF	FE	9E	63	32	F2	CE	23	6C	13	64	23	2E	00	8D
03A0	A6	CF	FD	9E	63	00	13	96	A6	AE	F2	CE	96	AB	30	00
03B0	13	DE	E1	23	E1	23	3F	00	13	DE	63	FF	4B	00	13	C6
03C0	63	01	3B	03	00	78	83	13	64	3B	84	34	4F	00	00	78
03D0	E1	23	E3	23	80	F0	81	30	A6	AE	F2	BE	00	78	64	FF
03E0	F4	EC	64	0A	65	04	04	10	F5	E1	F5	DE	35	00	13	EA
03F0	74	FF	34	00	13	E4	F4	EC	E1	23	15	1A	00	78	A6	95

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0400	FC	88	FC	88	7C	01	F2	CE	E1	23	62	1D	63	1D	13	D2
0410	F8	07	BE	F8	01	B9	F8	FF	AE	A9	EE	0E	FB	FF	73	29
0420	99	3A	1B	D4	6B	00	5B	03	00	78	6B	03	5B	03	00	78
0430	6B	0F	5B	03	00	78	6B	0A	5B	03	00	78	6B	EE	00	78
0440	96	AB	A6	CC	F2	CE	30	00	E1	21	3B	0F	14	78	63	08
0450	F3	EC	A6	AE	F2	CE	63	FF	83	23	14	66	04	80	73	FF
0460	43	05	14	76	E1	21	E1	31	82	30	80	F0	4F	00	14	5C
0470	63	20	F3	EC	14	7A	E1	21	60	00	A6	CC	F2	BE	00	78
0480	00	D4	4D	00	14	B4	C3	07	84	30	83	33	84	33	A6	B1
0490	F4	88	00	8D	F2	CE	31	00	14	B2	A6	7F	F3	88	F2	CE
04A0	00	94	96	9C	E1	23	80	F0	F2	BE	23	9C	7D	FF	23	9C
04B0	14	B4	00	94	63	00	65	00	A6	B1	00	8D	F2	CE	41	00
04C0	14	EA	75	01	96	9C	42	1E	96	9F	30	00	15	02	E1	23
04D0	E1	23	3F	00	15	02	42	1E	15	12	64	01	84	23	E1	23
04E0	42	1D	96	9F	E1	43	80	F0	82	40	00	94	F2	BE	73	01
04F0	33	08	14	BA	45	00	00	78	A6	C9	F2	CE	41	00	6C	03
0500	00	78	64	1A	84	24	4F	00	15	12	23	8C	7E	01	23	8C
0510	96	9C	E1	23	60	00	61	00	14	EA	4C	02	6C	03	3C	03
0520	13	FE	00	78	A6	C9	F2	CE	31	00	15	4E	4A	00	15	34
0530	7A	FF	00	78	A6	8F	C3	03	43	03	63	01	F3	88	F3	88
0540	F2	CE	96	A8	E1	23	80	F0	A6	C9	F2	BE	00	78	A6	C9
0550	F2	CE	A6	A2	96	A2	63	01	83	23	64	06	84	22	F4	88
0560	00	7F	64	06	84	32	F4	88	30	00	15	86	E1	23	E1	23
0570	3F	00	15	86	42	1D	15	86	E1	23	D1	33	80	F0	82	30
0580	A6	C9	F2	BE	00	78	E1	23	63	1A	83	24	3F	00	15	98
0590	A6	AE	60	80	F0	BE	15	9E	23	8C	7E	05	23	8C	60	00
05A0	61	00	CA	1F	15	80	24	24	23	A8	4C	03	16	6A	24	24
05B0	24	40	24	24	23	A8	4C	03	16	6A	24	82	4C	03	16	6A
05C0	24	24	23	A8	4C	03	16	6A	25	24	15	A6	A6	DA	F4	CE
05D0	85	E0	85	44	4F	00	16	16	85	E0	85	34	3F	00	15	E4
05E0	84	E0	16	12	84	30	85	E0	85	24	3F	00	15	F2	83	E0
05F0	16	12	83	20	85	E0	85	14	3F	00	16	00	82	E0	16	12

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0600	82	10	85	E0	85	04	3F	00	16	0E	81	E0	16	12	81	00
0610	80	E0	A6	DA	F4	BE	00	2D	23	38	60	01	61	FF	F1	E1
0620	24	24	4B	0A	00	78	F1	DE	31	00	16	20	40	00	00	78
0630	70	FF	16	1C	61	20	13	04	26	34	C9	07	A6	5A	F9	88
0640	F0	CE	8B	00	23	A8	24	40	24	82	23	A8	25	24	4C	03
0650	00	78	24	24	4B	0A	00	78	16	3A	00	00	03	00	03	03
0660	0F	03	00	00	00	03	00	0F	03	03	60	00	16	1C		
0670																
0680	07	06	0C	06	11	06	16	06	23	06	28	06	2D	06	32	06
0690	01	06	1D	06	39	06	22	00	18	00	1D	1D	10	28	10	92
06A0	FE	00	78	BC	78	DC	78	EC	78	F4	78	20	F8	F8	00	00
06B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
06C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
06D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
06E0	23	02	25	CC	4B	0A	16	F6	00	2D	26	38	4B	0A	16	F6
06F0	26	16	3B	0A	16	E8	00	2D	26	34	25	A6	16	E2		

AN INEXPENSIVE WIRING PENCIL

- by Dick Thornton, 1403 Mormac Rd., Richmond, Va. 23229 (USA)

When I first decided to use the solder-thru wiring pencil method for circuit construction, I visited a local store to purchase one of the Vector wiring pencils. My Scotch blood balked at spending $10 for a cone-shaped piece of plastic, however, so I decided to try making one at home. The result was very satisfying to me, and the drawing shows my approach.

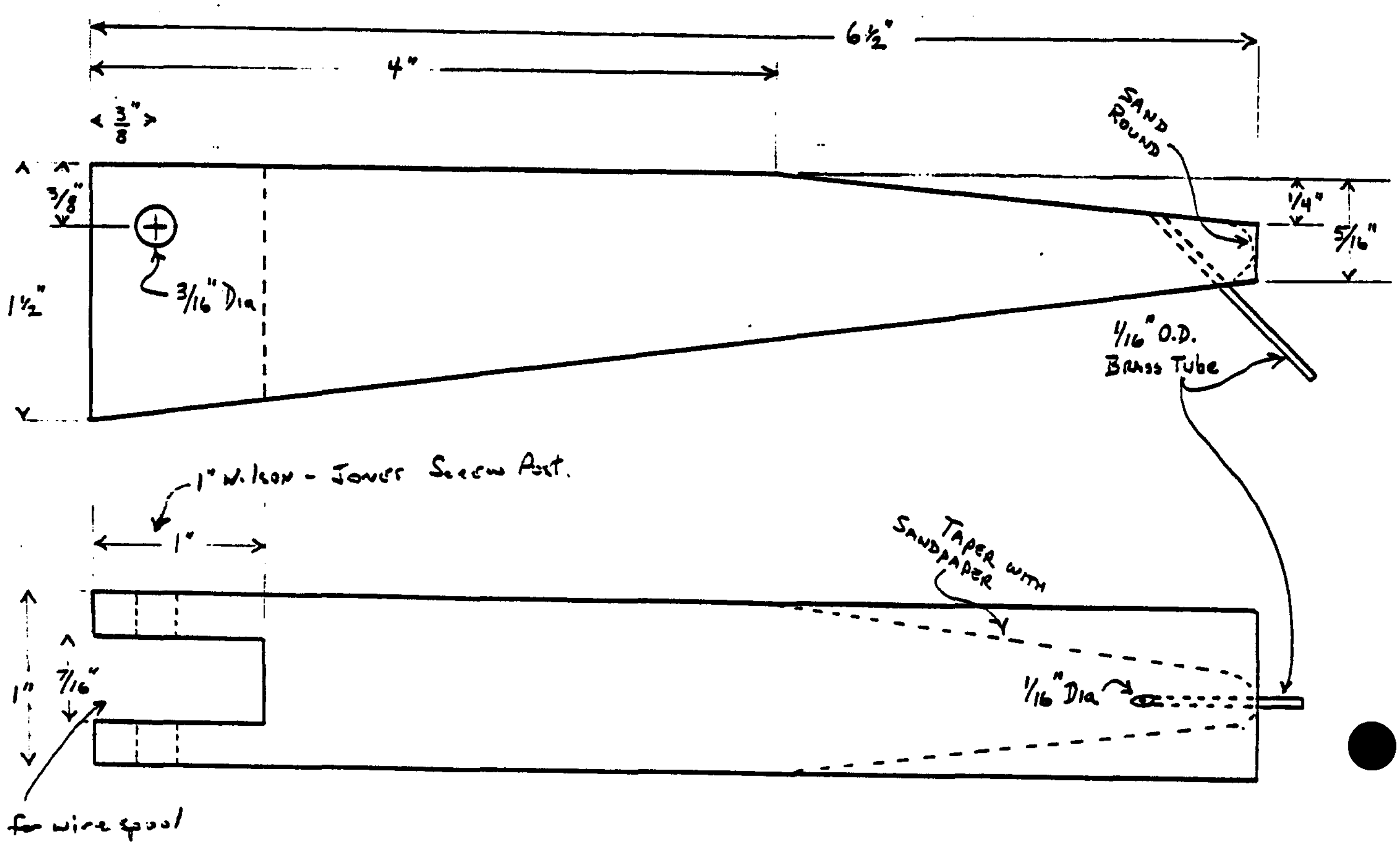
First, find a block of wood at least one full inch thick and cut out the shape in the top drawing. Drill the 3/16" hole for the screw post, and a 1/16" hole for the brass tubing.

Before cutting the 7/16" cutout for the wire spool shown at left in the second drawing, it is best to have a spool available so you can use it to test for fit. The cutout should be barely wider than the spool. Vector sells spools of wire as W32-9DP for 32 gage wire, and W36-9DP for 36 gage wire, or you can wind your own spools using sewing machine bobbins and solder through wire. Beldon sells this wire in a 1/2 pound spool with stock number 8056, if you can find a supplier.

Now sand the wood block so that all the sharp edges are smoothly rounded, and the front comes to a rounded conical point. Leave enough wood around the 1/16" hole to support the brass tube, though. I used a rotating disk sander on a drill with coarse sandpaper, then hand sanded with medium sandpaper. Later, I added indentations for my thumb and middle finger, and the result is a very comfortable tool. I finished by rubbing it with oil.

The axle for the wire spool is a screw post, available from office supply and stationery stores. Get the 1" length. To mount a wire spool, place it the cutout, put the long end of the screw post into one of the 3/16" holes, then screw the threaded end into the 3/16" hole on the opposite side. Brass tubing can be obtained from hobby stores. Cut a piece long enough for about 3/4" extending away from the front of the tool, smear a little glue on it, and insert it into the hole at the front. File the tubing flush with the top of the tool. File any burrs at the tip of the tube. Finally, twirl the tip of a knife in both ends of the tube to be sure there is a smooth path for the wire. You will be pulling wire through the tubing, and don't want to scrape insulation off the wire.

Thread the wire from the spool, along the top of the tool, and down through the tubing. When using the tool, wrap the wire around the pin to be wired, pressing the wire between your index finger and the top of the tool to maintain tension. Be sure to use a soldering iron with 750-850 degree tip temperature with this wire.



VDU - 126 x 64 Graphics Dump

- by George Musser, 60 Broadway Road, Warren, N.J. 07060

The following is a short, simple program which dumps the ACE VDU memory (128 x 64 mode) to an Epson MX-80 III printer. The output fills approximately 31 cm by 19 cm, with the x-axis running lengthwise. Since the MX-80 III can print a 480 dot graphics line, the program "expands" each bit of screen memory to a 7x7 dot matrix on the printer. While writing this program, I encountered problems with line spacing control; the correction may make the output seem slightly uneven in darkness.

My printer interface uses EF1 and output port 2. Note also that, in my printer, switch SW2-3 is set, to give automatic line feed with carriage returns. The program may be executed at any page boundary with P=0.

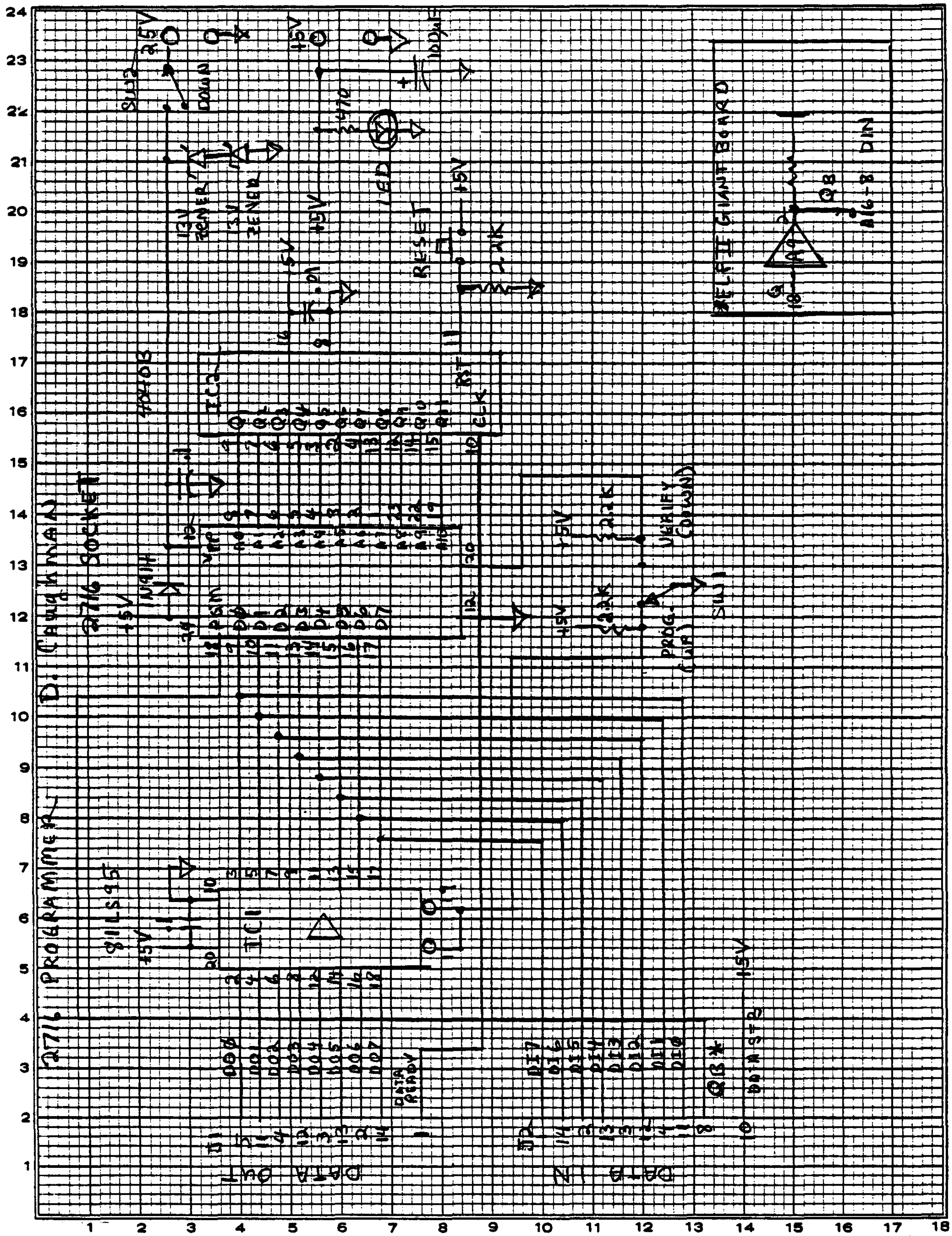
```
<XX> 00   90 B1 B2 F8 58 A1          init. printer PC
     06   F8 FF A2 E2                init. stack pointer
     0A   F8 10 AD                   init. counter
     0D   F8 1B D1 F8 31 D1          send line spacing
     13   2D F8 01                   begin graphics loop
     16   AF
     17   F8 E0 BC 8D AC F8 40 AE    init. memory pointer
     1F   F8 0D D1 F8 1B D1          send graphics mode
     25   F8 4B D1 F8 C0 D1
     2B   F8 01 D1
     2E   0C 52 8F F2 32 37          check if bit 1 or 0
     34   F8 FF C8
     37   D1 D1 D1 D1 D1 D1 D1       output bit
     3E   F8 10                      increase memory pointer
     40   1C FF 01 3A 40
     45   2E 8E 3A 2E                check if bit column done
     49   8F FE 3A 16                check if byte column done
     4D   8D 3A 13                   check if rows done
     50   F8 1B D1 F8 40 D1          clear printer
     56   00                         idle
     57   D0                         return
     58   3C 58 52 62 22 30 57       output byte to printer
```

A TWO CHIP EPROM PROGRAMMER FOR THE ELF

-by D. Caughman, 3795 Somerset Dr. S.W.
Marietta, Georgia 30064

The beauty of this programmer is its simplicity! There are only two IC's required and no tricky one-shots to worry about having to adjust. After all with a crystal controlled clock on the 1802 why not do it with software ? (Not everyone owns an oscilloscope!). The circuit was designed to work with a ELF II with a Giant Board, but if you have an input and output port on your computer it should not be much trouble to modify it to work; however, the software will have to be modified for clocks of different frequencies.

Connectors J1 and J2 connect directly up the DIN and DOUT port of the Giant Board. Add a jumper wire from A9-2 to the previously unused pin on the DIN socket A16-8 to bring out a buffered Q (Qb). The two series connected 13 volt zener diodes help clamp any >26 volt transients that may occur from the switching of SW2. Fellow ACE member Byron Bledsoe suggested this addition to me when certain brands (but not all) of 2716's would self destruct. Since the addition of the diodes I have not lost a single EPROM in over a year of use. So if you have tried building your own and had a simular problem this might be all you need to get it going. A 74LS244 could be substituted for the 81LS95 but the pinouts will have to be changed. The 4040 address counter is incremented everytime an OUT 7 instruction is executed. Since this will increment the address on the first byte, the program writes (reads) address 401H first then comes back to write (read) address 400H in the computers memory. This works because the counter will have overflowed causing 0000 to appear on the EPROMS address lines. The circuit can easily be modified to accomodate 2732's. The program runs at 0300H with the data loaded at 0400H to 0BFFH.
Once the routine and programming data have been entered using your monitor, then the HEX keyboard is to be used. Since the Q line is used for the programming pulse, any serial I/O using the Q line must not be used after the EPROM is installed in the socket with power on. (That is why the keypad is used).
....


2716 PROGRAMMER
D. Caughman
2716 SOCKET
81LS95
4040B
IC1
IC2
RESET
LED
ELF II GIANT BOARD
DATA OUT
DATA IN
VERIFY (DOWN)
PROG (UP)
13V ZENER
+5V
+25V

```
>GO 01F00
#
0000                              ORG #0300
0300  7A     START:  REQ
0301  F803           LDI #03
0303  BD             PHI RD
0304  F84B           LDI #4B ; GET PROGRAMMING ADDRESS
0306  AD             PLO RD
0307  F804           LDI #04
0309  BA             PHI RA  ; LOAD WRITE POINTER
030A  BB             PHI RB  ; LOAD READ POINTER
030B  F801           LDI #01 ; LOAD READ AND WRITE POINTER WITH
030D  AA             PLO RA ; LSB+1 SINCE HARDWARE COUNTER IS INCR.
030E  AB             PLO RB ; PRIOR TO READING OR WRITING
030F  F808           LDI #08   ; LOAD BYTE COUNTER
0311  BC             PHI RC
0312  F800           LDI #00
0314  AC             PLO RC
0315  F81F           LDI #1F  ;SET UP STACK POINTER
0317  B2             PHI R2
0318  F8FF           LDI #FF
031A  A2             PLO R2
031B  E2             SEX R2
031C  3F1C   H1:     BN4 H1 ; WAIT FOR INPUT
031E  371E   H2:     B4 H2 ; 07=READ 08=WRITE
0320  6C             INP 4
0321  64             OUT 4    ;DISPLAY IT
0322  22             DEC R2   ; RESTORE STACK POINTER
0323  F0             LDX
0324  FB08           XRI #08  ;READ OR WRITE ?
0326  3231           BZ OWRD
0328  FB0F           XRI #0F
032A  3231           BZ OWRD  ; CHECK FOR ILLEGAL CODE!
032C  E0             SEX R0
032D  64             OUT 4    ; EE=ERROR
032E  EE             BYTE #EE
032F  3000           BR START
0331  2C     OWRD:   DEC RC  ;DECREMENT COUNTER
0332  9C             GHI RC
0333  3A38           BNZ NCNT
0335  8C             GLO RC
0336  323B           BZ DNE
0338  DD     NCNT:   SEP RD ;START PROGRAM MODE
0339  3031           BR OWRD
033B  F804   DNE:    LDI #04
033D  BA             PHI RA   ; DONE NOW GO BACK AND PROGRAM
033E  BB             PHI RB   ; OR READ THE FIRST BYTE
033F  F800           LDI #00
0341  AA             PLO RA
0342  BB             PHI RB ;THIS IS BECAUSE THE HARDWARE CNTR
0343  AB             PLO RB ; WILL OVERFLOW TO 000
0344  DD             SEP RD ;GO TO SELECTED MODE ONE MORE TIME
0345  E0             SEX R0
0346  64             OUT 4 ; OUTPUT COMPLETE FLAG 'CC'
0347  CC             BYTE #CC
0348         ;
0348         ;
0348
```

```
                 ;
0348  3000              BR START
034A  DO       EPGM:    SEP RO
034B  E2                SEX R2
034C  FO                LDX   ; CHECK FOR MODE (READ OR WRITE)
034D  F6                SHR
034E  3360              BDF READ
0350  EB       WRITE:   SEX RB ; WRITE MODE
0351  67                OUT 7
0352  F808              LDI #08
0354  BE                PHI RE
0355  F847              LDI #47   ; LOAD 50 MSEC TIME CONSTANT
0357  AE                PLO RE    ; IN DELAY COUNTER RE
0358  7B                SEQ       ; START PGM PULSE
0359  2E       LP1:     DEC RE
035A  9E                GHI RE
035B  3A59              BNZ LP1
035D  7A                REQ
035E  304A              BR EPGM
0360  EB       READ:    SEX RB   ; READ MODE
0361  67                OUT 7
0362  2B                DEC RB   ;RESTORE PTR
0363  6F                INP 7   ;
0364  1B                INC RB
0365  304A              BR EPGM
0367
```

STEP BY STEP PROCEDURE
FOR PROGRAMING AND VERIFYING

PROGRAMING EPROM

1. Place data to be programmed at 0400H to OBFF of the computer memory.
2. Load program and execute.
3. Verify SW1 up and SW2 (24 Volts) down.
4. Insert EPROM.
5. Apply +5 volts.
6. Set up the 25 volt supply, then set SW2 up (on).
7. Push counter reset button S3.
8. Enter "08" on the keypad, press input key and wait approximately two minutes for "CC" to appear.
9. Set S2 down to disconnect the 25V supply.

TO VERIFY

1. Load and execute program if not already done.
2. Place SW1 and SW2 in down position.
3. Apply +5 volts.
4. Press counter reset switch.
5. Enter "07" on the hex keypad, then press the input key. A "CC" should appear on the hex readout almost instantly. Data from the EPROM should now be at 0400H to OBFF in the computers memory.

Adding the 1861 Video Display to the ACE CPU Board

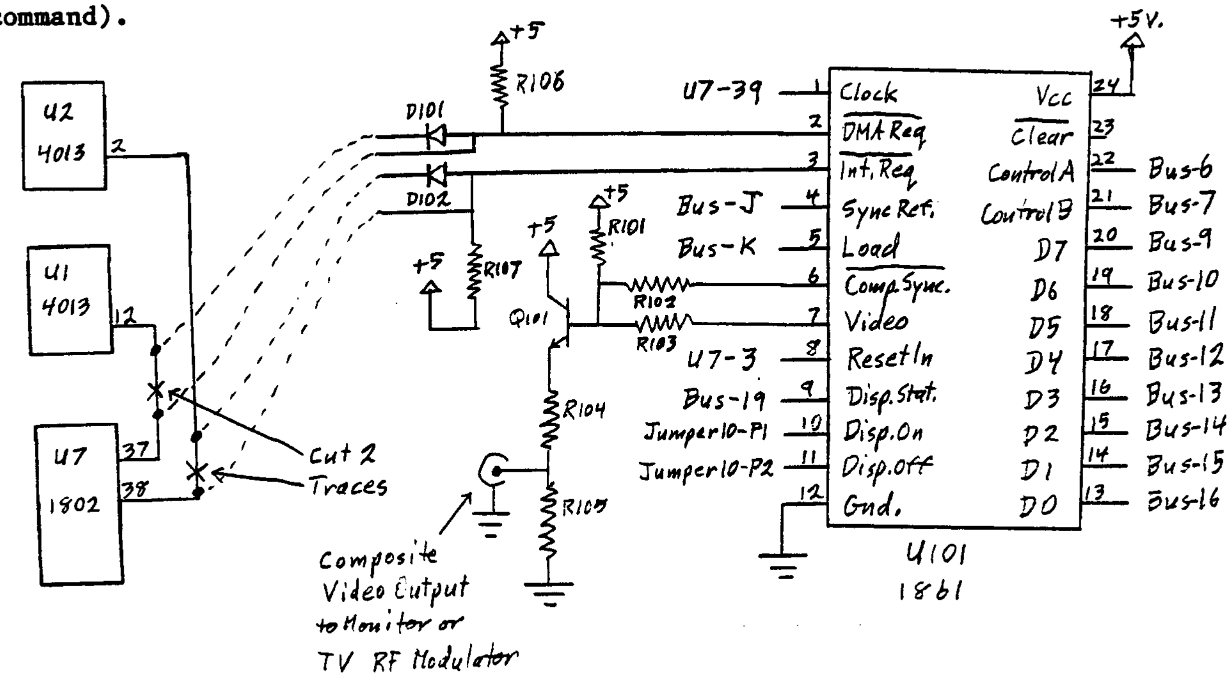
- by Larry Owen, 21A Regina Road, Trenton, Ontario. M8V 1G6

Although there are other, more powerful, video display controllers around, there are still those of us who like to play with the capabilities of the 1861. Considering its price and the amount of software available for it, I thought there might be others who are interested in how to hook it up on the ACE CPU card.

The following two diagrams show a schematic and a suggested parts placement guide. In order to differentiate the components in this circuit from those already on the board, I have added 100 to their numbers. The components required are:

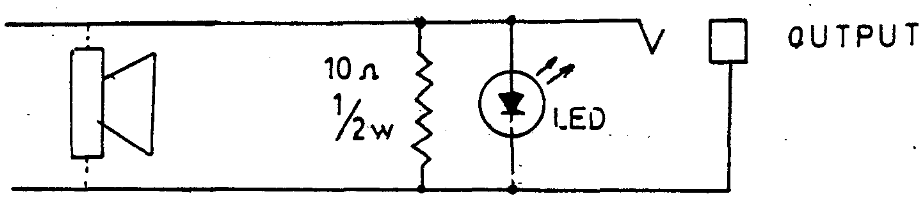
U101	1861
Q101	2N2222
D101	1N4148
D102	1N4148
R101	10K
R102	2K
R103	1K
R104	30
R105	200
R106	22K
R107	22K

These components are located in the breadboard area as shown, with the resistors mounted on the end. Most of the 1861's required connections can most easily be made at the board's buss connector, while a couple go directly to the 1802 CPU. Two more are shown going to Jumper 10. These select the N-Line Decoder outputs Port 1 (to enable the 1861) and Port 2 (to disable it). These, along with the Display Status signal going to the 1802's EF1 flag, will make this circuit compatible with software written for Quest systems, amongst others. Some people may wonder at the lack of isolation diodes from the 1861's DMA and Interrupt Request lines; they are not needed here because these 1861 outputs are from open drain transistors. If the 1802's EF1 line is ever required for another purpose, the 1861 will have to be removed from its socket, as it activates this signal even when it is turned off (by a software command).



Homebrew ELF Enhancements and a Mini Chip 8 Game
- by A. Boisvert, Quebec, P.Q.

After reading the first page of the last IPSO FACTO, I decided to contribute by doing this article. I have a homebrew ELF from Popular Electronics articles. I have replaced the data switches by a hex keyboard made of TTL ICs and diodes for the decoding circuit. (See diagram). I am using the cassette interface suggested by E. McCormick (P.E. Feb. '78). I have a cheap cassette recorder that works fine with this interface. I have removed the speaker from the recorder and replaced it with a 10 ohm 1/2 W resistor with a LED in parallel with it. The LED serves as an output indicator.



The following program is called HEX GAME, that Paul Moews wrote in one of his booklets. His mini CHIP-8 loads into memory from 0068 to 00FF. It uses 10 of the regular CHIP-8 instructions plus two others (keyboard read and write display). HEX GAME uses memory addresses from 0000 to 0063. I have included a description of the MINI CHIP-8 instructions.

The idea of the game is to have others try to guess a secret hex byte that you have entered in the computer. Each player is allowed five tries to guess the hex digits.

At the start of the program, enter the secret byte then the number of players. (Maximum is 09). The computer will then display 01 for the first player who will enter his selection and hit ENTER (IN switch). If he guesses the secret byte a tone will be heard to indicate he is the winner. If the value entered is not equal to the secret byte the ELF will display A0 (Value too HI) or BA (Value too LO), and then display a player number. If the secret byte is not guessed after five rounds, the computer will display it and restart at the beginning of the program, after generating a tone.

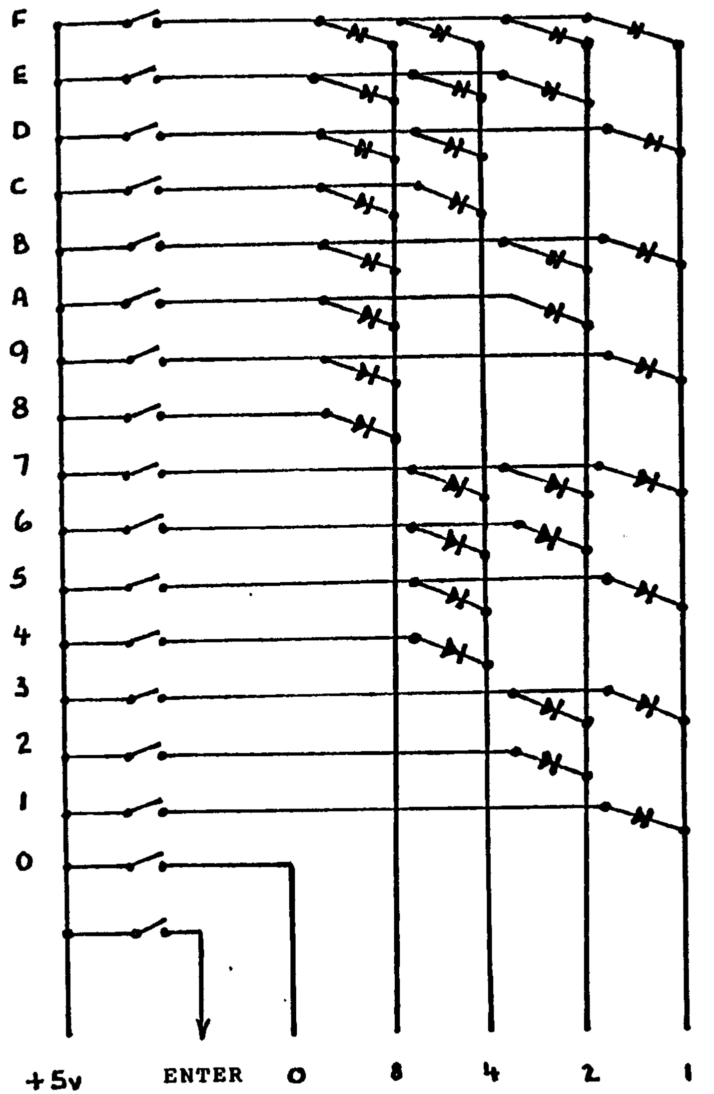
MINI CHIP-8 FOR ELF

MINI CHIP-8 INSTRUCTIONS

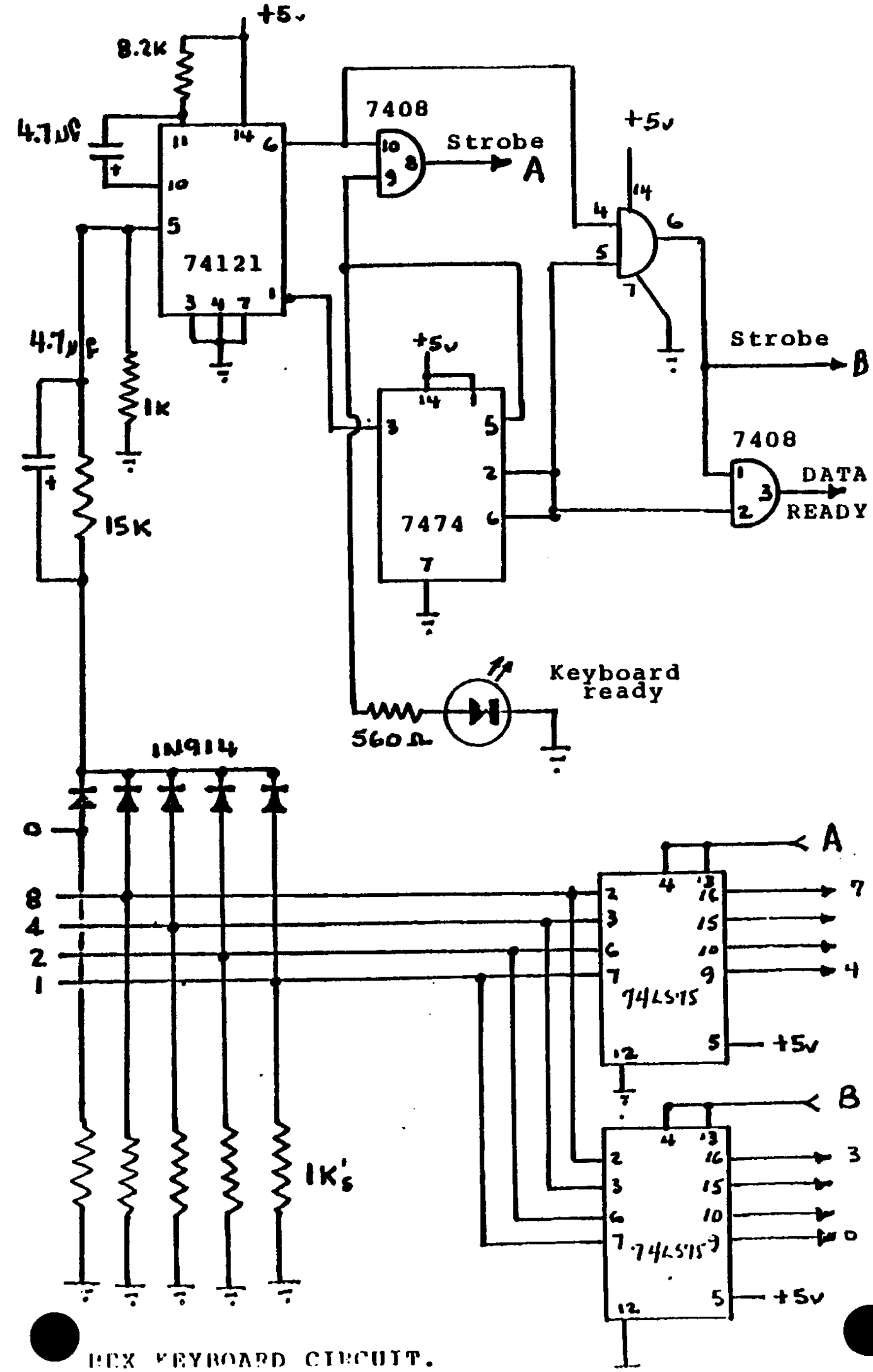
00MM	Do a machine code subroutine at location MM. (The machine code subroutine must end with D4)
10MM	Go to location MM in the program written in CHIP-8.
20MM	Do a CHIP-8 written subroutine at location MM. (The routine must end with 009E).
4XKK	Skip next program instruction if Vx does not equal KK. (KK = constant).
6XKK	Set variable Vx equal to KK
8XY0	Set variable Vx equal to Vy.
8XY1	Logically OR Vx &Vy. Result in Vx, (Vf is modified by this instruction).
8XY2	Logically AND Vx &Vy. Result in Vx. (Vf is modified by this instruction).
8XY4	ADD Vx & Vy. Result in Vx. (Vf is 00 if sum is less or equal to FF, 01 if sum is greater than FF).
8XY5	Subtract Vx & Vy. Result in Vx. (Vf is 00 if Vx is less than Vy, 01 if Vx is greater than or equal to Vy).
DXKK	Display Vx on hex display for KK time. (KK set to FF gives a 2 second display).
FX00	Read keyboard (switches) into Vx. Wait for the ENTER (IN SWITCH) to be pushed and released.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6-	00	00	00	00	00	00	00	00	F8	00	B2	B3	B4	B5	B6	B7
7-	BC	F8	68	A2	F8	7A	A4	F8	02	A5	D4	E2	45	AF	F6	F6
8-	F6	F6	32	98	F9	A0	AC	8F	F9	F0	A6	05	F6	F6	F6	F6
9-	F9	F0	A7	0C	A3	D3	30	7B	45	30	94	45	56	D4	42	A5
A-	D4	B5	B0	E5	B8	E5	9B	E5	C0	E5	E5	E5	E5	E7	E5	DD
B-	15	85	22	52	25	45	A5	D4	45	E6	F3	32	BF	15	15	D4
C-	45	FA	0F	3A	C8	07	56	D4	AF	22	F8	D3	73	8F	F9	F0
D-	52	F6	07	D2	56	F8	FF	A6	F8	00	7E	56	D4	7B	3F	DE
E-	37	F0	E6	6C	7A	45	D4	E6	64	45	BF	2F	9F	3A	EB	D4
F-	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

ADD.	CODE	DESCRIPTION
0000	3068	BRANCH TO INTERPRETER
0002	63A0	SET V3 TO A0
0004	64BA	SET V4 TO BA
0006	6AEE	SET VA TO EE
0008	6B09	SET VB TO 09
000A	6501	SET V5 TO 01
000C	6005	SET V0 TO 05
000E	F600	PUT SECRET BYTE IN V6
0010	FD00	PUT NUMBER OF PLAYERS IN VD
0012	DD40	DISPLAY IT
0014	8BD5	VB-VD TO FIND IF # OF PLAYERS IS VALID
0016	4F00	SKIP OVER IF VALID
0018	103C	GO DISPLAY EE, # OF PLAYERS OVER 09
001A	6C01	SET VC TO 01
001C	DC40	DISPLAY PLAYER #
001E	F100	PLAYER INPUT
0020	D140	DISPLAY VALUE ENTERED
0022	8165	V1-V6 TO FIND IF VALUE IS GUESSED
0024	4100	SKIP OVER IF NOT GUESSED
0026	1040	GO TO WIN TONE ROUTINE
0028	4F01	TEST VF TO FIND IF VALUE ENTERED IS HI OR LO
002A	1030	GO DISPLAY A0 VALUE TOO HI
002C	D440	DISPLAY BA VALUE TOO LO
002E	1032	PROCEED WITH NEXT PLAYER
0030	D340	DISPLAY A0
0032	8D55	DECREMENT # OF PLAYERS BY 1
0034	4D00	MORE PLAYERS LEFT
0036	1058	NO, TRY ANOTHER TURN
0038	8C54	YES, ADD 1 TO VC
003A	101C	NEXT PLAYER TURN
003C	DA20	DISPLAY EE (ERROR)
003E	1010	TRY AGAIN
0040	0044	BRANCH TO TONE SUBROUTINE (WIN)
0042	1008	RESTART GAME
0044	F8FFA8)	
0047	7A88A9)	
004A	2989)	
004C	3A4A)	TONE ROUTINE TO INDICATE A WINNER
004E	3147)	
0050	7B2888)	
0053	3257)	
0055	3048)	
0057	D4)	
0058	8DC0	SAVE # OF PLAYERS
005A	8055	SUB. # OF ROUND BY 1
005C	4F01	MORE ROUND LEFT
005E	101A	YES, GO TO PLAYER #1
0060	D640	DISPLAY SECRET BYTE
0062	1040	GENERATE TONE



HEX KEYBOARD.



HEX KEYBOARD CIRCUIT.

AN 1861 TVT FOR FORTH

-by David Ruske, R2 Box 250, Waupun, Wi, USA 53963

I was a little frustrated. . .just when I had my ACE dynamic board going with enough memory to run FORTH, my TVT board decided to go on the fritz. (Anyone know where I can get a 9324 without a $50 minimum order)? Not having a copy of Tom Pittman's DOTS program, I decided to write my own. The display is 16*16 (funny looking but readable), and features carriage return, backspace, home/clear, and automatic scrolling. Register usage does not conflict with FORTH, and Registers 4, 5, and 6 are left free for SCRT. The program itself occupies 1 page, the dot table takes 1 page, and 4 pages are used for display. I have the program located at page EA; for a different location modify the underlined bytes. Credit for the dot table goes to T. Cruvisson, TVT for the 1861, IPSO #17. You must initialize RE to a spare location , I used EAFE. No value is assumed on entry. Additionally, your FORTH initialization code should initialize R2 to the return stack location (see FORTH Implementation Notes by Tony Hill, IPSO #29) for the benefit of the interrupt routine, and R1 should be initialized to the interrupt routine (EAEC in this listing). Lastly, it should turn on the 1861 (E2 69 on ACE systems). Note that while FORTH is running the display may jitter, since FORTH uses several 3-cycle instructions. The display is stable while FORTH is waiting for an input, etc. For this listing, the entry point for EMIT is EA00, and CR enters at EA0C.

Closing comments: thanks to anyone involved in putting out the ACE dynamic board! Tony Hill: thanks for creating a program like window. I anxiously await PEEPHOLE.

```
EA00   19 09 A7        (Get char. from FORTH, put in R7.0)
  03   29 29 29        (Clean up R9 for FORTH)
  06   EE              (X=RE)
  07   87 FB 0D        (Is it CR?)
  0A   3A 2E           (If so,)
  0C   EE              (   Make sure X=RE again for CR entry)
  0D   8F FF F0        (   Is position F0?)
  10   33 C9           (   If so, go to scroll)
  12   8F FA F0        (   Else mask off lo nibble)
  15   FC 10           (   Add 10 (all #s in hex))
  17   AF              (   And replace the position pointer)
  18   FA C0           (   Get bits 6 and 7)
  1A   7E 7E 7E        (   Shift to bits 0 and 1)
  1D   EE              (   NOP)
  1E   FC EC B8        (   Add to disp. page start, put in R8.1)
  21   8F FA 0F        (   Get lo nibble)
  24   F6 5E           (   Shift right and store it)
  26   8F FA F0        (   Get hi nibble)
  29   FE FE           (   Shift it left twice)
  2B   F4 A8           (   Add to stored byte and put in R8.0)
  2D   DC              (   Return to FORTH)
  2E   87 FB 08        (Is it Backspace?)
  31   3A 38 8F        (If so,)
  34   32 37           (   Return if position is 00)
  36   2F              (   Else decrement position and)
  37   DC              (   Return to FORTH)
```

```
EA38   87 FB 0C              (Is it Home?)
  3B   3A 50                 (If so,)
  3D   F8 00 AF              (   Force position to 00)
  40   F8 FF A8              (   R8 will be pointer tc disp. page)
  43   F8 EF B8              (   EFFF is hi byte of disp. pages)
  46   F8 00 58 28           (   Zero the byte and decrement pointer)
  4A   98 FB EB              (   Is pointer hi=page below display?)
  4D   3A 46                 (   If not, go back and do it again)
  4F   DC                    (   Return to FORTH)
  50   87 FA 01              (Assume valid char...is it odd?)
  53   32 56 7B              (If so, set Q)
  56   87 FF 20 A7           (Subtract 20 from char.)
  5A   FA 0F                 (Get the lo nibble of this)
  5C   F6 5E                 (Shift it right and store it)
  5E   87 FA F0              (Get the hi nibble of this modified byte)
  61   FE FE                 (Shift it left twice)
  63   F4 A7                 (Add it to lo nibble and keep it)
  65   F8 EB B7              (R7 is now dot table pointer)
  68   8F FA C0              (Get bits 6 and 7 of position)
  6B   7E 7E 7E              (Shift to get these in bits 0 and 1)
  6E   EE                    (NOP)
  6F   FC EC B8              (Add to disp. page start)
  72   8F FA 0F              (Get lo nibble of position)
  75   F6 5E                 (Shift it right and store it)
  77   8F FA F0              (Get hi nibble of position)
  7A   FE FE                 (Shift it left twice)
  7C   F4 A8                 (Add it to lo nibble and keep it)
  7E   8F BF                 (Preserve position byte)
  80   F8 08 AF              (So R8 can count to 8)
  83   9F FA 01              (Is position even?)
  86   32 96                 (If not,)
  88   31 91                 (   Is char. even?)
  8A   07 F6 F6 F6 F6        (   If so, shift right four times)
  8F   30 A2                 (   and continue)
  91   07 FA 0F              (   else get lo nibble of table byte)
  94   30 A2                 (   and continue)
  96   31 9D                 (If position and char. are even)
  98   07 FA F0              (   Use hi nibble from table)
  9B   30 A2                 (   and continue)
  9D   07 FE FE FE FE        (Position is even, char. is odd)
  A2   5E                    (Continue here; store prepared byte)
  A3   9F FA 01              (Position odd?)
  A6   3A AF                 (If not,)
  A8   08 FA 0F F1 58        (   Mask off hi of prev. byte and OR)
  AD   30 B4                 (   and prepare for next loop)
  AF   08 FA F0 F1 58        (   else mask lo and OR it in)
  B4   88 FC 08 A8           (Add 8 to Memory pointer)
  B8   87 FC 08 A7           (Add 8 to Table pointer)
  BC   2F 8F                 (Decrement count)
  BE   3A 83                 (If 0, continue, else loop)
  C0   7A                    (Kill Q for next time)
  C1   9F AF                 (Restore position)
  C3   FB FF                 (Is position FF?)
  C5   32 C9                 (If not,)
  C7   1F                    (   increment position)
```

```
EAC8  DC            (Return to FORTH)
  C9  F8 F0 AF      (New position=F0)
  CC  F8 EC B7 B8   (EC is lowest display page)
  D0  F8 40 A7      (R7 is source byte for move (scroll))
  D3  F8 00 A8      (R8 is destination byte)
  D6  07 58         (Get source byte, store at destination)
  D8  17 18         (Increment source and destination)
  DA  97            (Get R7.1)
  DB  FB F0         (Is it above display area (F0=disp.+1))
  DD  3A D6         (If not, loop)
  DF  27            (R7 is now EFFF)
  E0  F8 00 57      (Erase byte)
  E3  27            (Set next byte up)
  E4  87 FB BF      (Line erased yet?)
  E7  3A E0         (If not, loop)
  E9  DC            (Otherwise return to FORTH)
  EA  72 70         (Interrupt routine)
  EC  C4 22 78      (Initialize R1 here (EAEC))
  EF  22 52 E2 E2
  F3  F8 EC B0
  F6  F8 00 A0
  F9  30 EA
```

Dot Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
EB00	04	A0	A0	62	28	00	00	00	04	AA	E0	62	44	44	00	00
10	04	0E	86	20	44	A4	00	02	04	0A	E2	40	44	4E	06	04
20	04	0E	24	A0	44	A4	00	08	04	0A	E8	A0	44	44	40	40
30	04	00	AC	40	28	00	40	40	00	00	00	00	00	00	80	00
40	44	EE	2E	4E	EE	00	00	0E	AC	A2	68	A2	AA	00	20	8A
50	A4	22	A8	82	AA	22	46	42	A4	E6	EC	C4	EE	22	80	24
60	A4	82	22	A8	A2	00	46	44	A4	82	22	A8	A2	22	20	80
70	4E	EE	2C	48	E2	22	00	04	00	00	00	00	00	04	00	00
80C	04	CE	CE	E6	AE	2A	8A	AE	EA	A8	A8	88	A4	2A	8E	AA
90	EA	A8	A8	88	A4	2C	8E	EA	AA	E8	AC	C8	E4	28	8E	EA
A0	EE	A8	A8	8A	A4	2C	8A	EA	EA	A8	A8	8A	A4	AA	8A	AA
B0	4A	CE	CE	86	AE	4A	EA	AE	00	00	00	00	00	00	00	00
C0	C4	CE	EA	AA	AA	E6	0C	00	AA	AA	4A	AA	AA	24	04	40
D0	AA	A8	4A	AA	AA	24	84	A0	CA	CA	4A	AE	44	44	44	00
E0	8A	C2	4A	AE	A4	84	24	00	84	AA	4A	AE	A4	84	04	00
F0	82	AE	4E	4A	A4	E6	0C	00	00	00	00	00	00	00	00	0E

ADDITIONAL NOTES:
1861 TVT FOR FORTH

There is a small problem in using an interrupt routine with FORTH. When the interrupt occurs, the stack pointer (R2) is decremented to the location where the main program's PC and stack pointer will be stored. Thus, the main program (in this case FORTH) must never have R2 incremented above valid data, which would be lost if the interrupt occured at that point. The only problems I have encountered so far involve the words LOOP, +LOOP, and I. Tony Hill told me how to fix the loop words LOOP and +LOOP, and these fixes should appear in one of his more recent articles. The I word was fixed as follows:

```
1425  92 B8 82 A8
  29  18 19 19 19
  2D  48
  2E  C0 XX XX

XXXX  59 29 08 59 DC
```

The bytes XX XX may be any spare five byte location in memory, where the remainder of the code is stored. You may wish to locate this after your I/O routines or between your initialization code and FORTH.

The following routines may come in handy when using this TVT:

```
HEX
: HOME 0C EMIT ;
: INVERSE EC00 EB00 DO I DUP C@ FF XOR SWAP C! LOOP ;
     (This routine gives inverse video the first time the word
      is invoked,, normal the second time,,etc.  Substitute the
      end of the shape table page + 1 and the beginning of the
      shape table page for EC00 and EB00 respectively.)
CREATE TVOFF EE C, 61 C, 2E C, DC C, SMUDGE
CREATE TVON EE C, 69 C, DC C, SMUDGE
     (Turning the TV off during calculations will improve speed
      ---the 1802 won't spend half its time refreshing the display.)
```

Nies Text Editor Modifications

- by George Musser, 60 Broadway Road, Warren, N.J. 07060

I have made several modifications to Steve Nies' Text Editor in order to increase its usefulness as word processing software. The original program listings of The Monitor version II (SYSMON) and The Text Editor (SCRIPTORY) may be found in *Ipso Facto* issues 20 and 23. I give all due credit to Steve for two excellent programs.

The first set of patches allows an emulation of upper/lower case characters using the 6847. Upper-case characters are now displayed as inverse, and lower-case as normal. Required changes are to OUTCHAR and the monitor MAIN BODY. A new INCHAR routine (which flashes the cursor) is supplied as well as an additional OUTCHAR routine.

A second patch allows faster entry of text by, when the right side is reached, simply scrolling the screen halfway over. Previously, the screen scrolled only one column, thus making text entry very slow.

A third series of changes helps accomodate printer control codes with a new EDIT subcommand, Escape. Press ESC and the desired control code; this code will be stored directly in memory. Control codes print as the inverse of the corresponding character between 20 (SP) and 3F (?); for example, control-Q prints as inverse "1".

The fourth routine is an additional Text Editor command:

15) FORMAT AA BB CC

This command allows a simple formatting option: margins and line spacing. Specify the left and right margins in hex as AA and BB respectively; CC determines line spacing. The routine will enter the Get parm mode and wait for you to indicate the block of memory to be formatted. Briefly, this program works by removing all carriage returns and then replacing them according to the desired margins. When the break between lines occurs in the middle of a word, the routine will stop and allow you to specify hyphenation; type in the characters to be left on the upper line, or hit the carriage return for words not to be hyphenated. Memory locations S 9A, 9B, and 9C are used for storage of values.

<X1> and <X2> are simply any two new pages.

LISTINGS

(i) Changes to MAIN BODY (The Monitor version II):

```
I 38    C0 <X1> 00 C0 P 65            new I/O vectors
```

Changes to OUTCHAR:

```
P 65    73                            entry - save char.
  78    08 FB 80 58                   turn cursor off
  7F    C0 <X1> 36                    branch to char. convert
  AA    08 FB 80 58                   turn cursor on
  C1    42 B8 02                      restore R(8).1, char.
```

New INCHAR routine: uses EF3 and input port 7

```
<X1> 00   97 73 87 73 8F 73           save R(7) and R(F)
     06   F8 S  BF F8 95 AF           init. R(F)
     0C   4F B7 0F A7                 get cursor location
     10   07 73                       save character
     12   F8 18 BF 07 FB 80 57        flash cursor
     19   2F 9F 32 12
     1D   3E 19 6F                    check if key pressed
     20   F8 FF AF                    do debounce delay
     23   2F 8F 3A 23
     27   36 12                       check for bounce
     29   12 42 57 42 AF 42 A7 02 B7  restore char., R(F), R(7)
     32   6F FF 00 D5                 return
```

— adapted from Tony Hill's Window

Character conversion routine:

```
<X1> 36    CB S  6A                   check if control char.
     39    FF 21 3B 44                check if upper-case
     3D    FF 1A 33 44
     41    FC DB BF
     44    9F FF 61 3B 50             check if lower-case
     49    FF 1A 33 50
     4D    FC 5B BF
     50    9F FA BF C0 P  83          return
```

(ii) Modification to CURSOR RIGHT:

```
R F4   C0 <X1> 56                     branch to patch
```

New Routine:

```
<X1> 56    F7 CB R  F7                check if past edge
     5A    8F FC 10 AF                increase scroll counter
     5E    8E FF 0F AE                adjust screen position
     62    D4 Q  EE                   display new screen
     65    C0 R  BD                   continue
```

(iii) Modification to INSERT CHAR:

```
V AD  CA <X1> 68                      branch if not control-C
```

New routine:

```
<X1> 68    9C FB 1B CA R  BD          check for Escape
     6E    D4 S  64 3B 6E             wait for keypress
     73    FF 20 C3 R BD              check if control char.
     78    9F 5D FC A0 C0 V 83        store, print char.
```

Modification to SCREEN PRINT:

```
R 5B   C0 <X1> 7F                     branch to patch
```

New routine:

```
<X1> 7F    FF 20 CF FC A0 FC 20       check for control char.
     86    D4 S  67 D5                print appropriate char.
```

(iv) Modification to command table:

```
Q E3   01 <X2> 00                     command table extension
```

New command table entries:

```
<X2> 00    43 48 00 Y 87              (Change)
     05    4F 00 <X1> 0C              (Format)
     09    FF 00 00                   reserved for table extension
```

New routine:

```
<X2> 0C    D4 <X2>  D6 A1 D4 <X2>  D6 A0 get margins
     14    D4 <X2>  D6 3B D3 FF 01       get line spacing
     1B    DA 38 80 DA 36 81 DA 34       store values
     23    F8 05 D4 Ä  45 33 D3          get limits
     2A    9C BE 8C AE                   prepare parameters
     2E    D4 I  DD 8C                   print Form Feed
     32    9D B1 8D A1                   save starting address
     36    D4 R  A2                      find CR/FF
     39    D4 I  F7 33 4A                check if finished
     3E    1D 0D FB 20 32 36             check if next byte is space
                                         (to maintain paragraphs)
     44    2D F8 20 5D 30 36             change CR/FF to space
     4A    91 BD 81 AD 38                restore starting address
     4F    1D                            increment pointer
     50    DA 35 A7 F8 20 D4 <X2>  DE    create left margin
     58    DA 37 A0                      store line length
     5B    D4 I  F7 33 AC                check if done
     60    D4 R  8B 32 4F                check for CR/LF/NUL
     65    1D 20 80 3A 5B                check line length
     6A    0D FB 20 32 A7                check if space
     6F    2D 10 0D FB 20 3A 6F          backspace to first blank
     76    9D 73 8D 73 1D                save location
     7B    4D D4 S  67 20 80 3A 7B       print word
     83    D4 I  DD 3F A0                print question mark
     88    F8 00 A9 D4 J  05             get hyphenation
     8E    12 89 F4 AD 12 02 7C 00 BD    hyphenate word
     97    89 32 A7 1D                   check if hyphen to be stored
     9B    F8 02 A7 F8 20 D4 <X2>  DE    make room for hyphen
     A3    F8 2D 5D 1D F8 0D 5D 30 4F    store hyphen and CR
     AC    F8 01 A7 F8 0D D4 T  B3       end of CR insertions
     B4    91 BD 81 AD                   restore starting address
     B8    D4 R  A2 1D                   find CR/FF
     BC    D4 I  F7 C3 W  D6             check if done
     C2    DA 39 A7 F8 0D D4 <X2>  E7    insert new line spacing
     CA    DA 39 1D FF 01 3A CC 30 B8
     D3    C0 M  46                      parameter error
     D6    F8 02 B7 D4 J  AE 8C D5       get value
     DE    87 52 8E F4 AE 9E 7C 00 BE    adjust R(E)
     E7    9F D4 T  B3 D5                call MOVE LINE
```

RELOCATE

by M.E. Franklin, 690 Laurier Ave., Milton, Ont. L9T 4R5

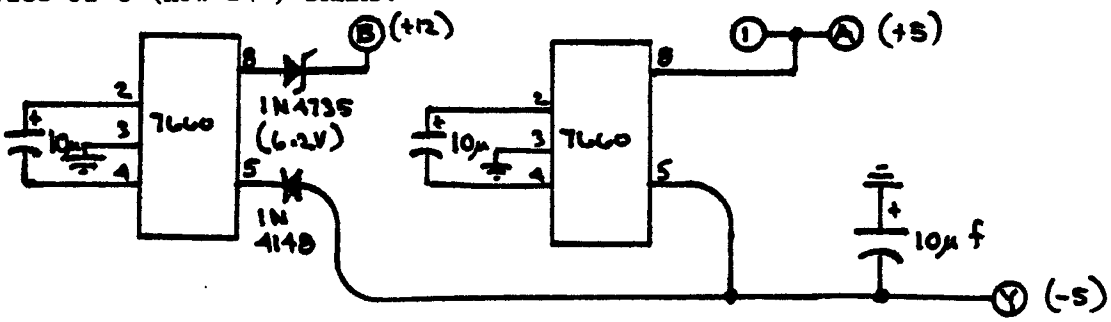
This program is designed to facilitate writing hex code in RAM for later relocation to EPROM or or another address location. The program steps through a source code looking for SCRT calls (04) or long branches (C0, C2, CA) and adds a preset off set to the page value when found, providing certain conditions are met, ie. that it is not a stack page, a monitor page etc. Of course, the program does not correct Load Immediate or calculated addresses. Even with its shortcomings, this is a handy piece of code for the HEX programmer.

```
1000  F810    LDI #10
1002  BA      PHI RA     Set length of source program in RA        Main program
1003  F800    LDI #00
1005  AA      PLO RA
1006  F800    LDI #00
1008  B1      PHI R1     Set start address of source in R1
1009  F800    LDI #00
100B  A1      PLO R1
100C  01      LDN R1     Load program byte
100D  FFD4    SMI #D4    Test if Call or longbranch instruction
100F  322B    BZ  #2B    Branch to fix routine if true
1011  01      LDN R1
1012  FFC0    SMI #C0
1014  322B    BZ  #2B
1016  01      LDN R1
1017  FFC2    SMI #C2
1019  322B    BZ  #2B
101B  01      LDN R1
101C  FFCA    SMI #CA
101E  322B    BZ  #2B
1020  11      INC R1     Inc source program counter                 Next byte
1021  2A      DEC RA     Dec count
1022  9A      GHI RA     Test if done
1023  3A0C    BNZ #0C    Loop if not
1025  8A      GLO RA
1026  3A0C    BNZ #0C
1028  C0FE00  LBR #FE00  Exit to monitor if true
102B  11      INC R1     Inc source program counter to page byte    Fix routine
102C  01      LDN R1     Load page byte                             Exclusions
102D  FFFE    SMI #FE    Test for exclusions - stack at FEFF
102F  3244    BZ  #44
1031  01      LDN R1
1032  FFC0    SMI #C0                        - first page of monitor
1034  3247    BZ  #47
1036  01      LDN R1
1037  FFC1    SMI #C1                        - monitor routine page
1039  3247    BZ  #47
103B  01      LDN R1
103C  FFC7    SMI #C7                        - monitor I/O page
103E  3247    BZ  #47
1040  01      LDN R1     Load page byte
1041  FC30    ADI #30    Add off set to page value                  Off set
1043  C8      LSKP       Skip
1044  F8FE    LDI #FE    Make stack page FE                         Stack fix
1046  51      STR R1     Store in program source
1047  11      INC R1     Inc source program counter
1048  2A      DEC RA     Dec length count twice
1049  2A      DEC RA
104A  3020    BR  #20    Branch to Next byte
83:05:01
20:46:38
```

Minus 5 Volts for the 64K Dynamic Board

- by Don Stewart, 3001 Fleet Street, Coquitlam, B.C.

I recently àssmebled the ACE 64K board, tried it with one 4116 chip and it worked; I put in the next 7 and they all worked; I shut off power to fix some details and destroyed the DRAMs by unplugging my supplies in the wrong order. True, they should all be on 1 power cord but the -5 VDC was a new supply.
So that I wouldn't do that again I have incorporated 2 Intersil 7660 Voltage Converters as shown to ensure that if either the +12 or +5 VDC is present I will have my -5 VDC. Also this is much cheaper than another supply. Really only one converter should be required, but the second one seemed cheap beside the price of 8 (now 24) DRAMs.



This fits very easily in the "kluge" area, and cost about $10.00. I would suggest that the 10 mfd. reserve cap be beside the 7660's, install the 10 mfd. cap which is part of the board design in addition to this one. Also, I should mention that the polarity mark on my PCB is reversed for the -5 VDC cap.

I have been most pleased with the board - lots of memory in little physical space, well thought out, good artwork and clear instructions - very well done.

****

1802 MICRO COMPUTER CONFERENCE

20 August, 1983

SHERIDAN COLLEGE, OAKVILLE ONTARIO

A full day of speakers and demonstrations on micro computer products, including the latest information on the RCA 1800 series products.
The program and events are oriented to hobbiests, and to industrialists who use micro processors for control or processing applications.

CLUB COMMUNIQUE

Rev. 83.04.01

NAME: ______________________ DATE: ______________

PRODUCT ORDER	QUANTITY	UNIT PRICE	TOTAL
CPU Board	______	$40.00	______
Backplane and I/O Board, Ver. 2	______	40.00	______
Front Panel (with EPROM Burner, Clock)	______	35.00	______
I/O Adapter for Backplane, Ver. 1	______	20.00	______
64K Dynamic (4116) Board	______	50.00	______
EPROM (2716/32) Board	______	40.00	______
Kluge (wire wrap) Board	______	25.00	______
Netronics - Ace Adapter Board	______	25.00	______
DMA Adapter Board (ELF II)	______	3.00	______
VDU Board, Ver. 2	______	40.00	______

Software

Fig FORTH - Netronics Cassette format (6K) 0000H	______	$10.00	______
Tiny Pilot - Netronics Cassette format (2K) 0000H	______	$10.00	______
SYMON - Netronics Cassette format (2K) C000H	______	$10.00	______

Back Issues

"Defacto" Year 1 - 3 (Edited)	______	$20.00	______
Year 4 Reprint	______	10.00	______
Year 5 Reprint	______	10.00	______

Membership

Current Year - Sept. '82 - Aug. '83
includes 6 issues of Ipso Facto

Canadian	______	$20.00 Cdn.	______
American	______	20.00 U.S.	______
Overseas	______	25.00 U.S.	______

PRICE NOTE

Prices listed are in local funds. Americans and Overseas pay in U.S. Funds, Canadians in Canadian Funds. Overseas orders: for all items add $10.00 for air mail postage. Please use money orders or bank draft for prompt shipment. Personal cheques require up to six weeks for bank clearance prior to shipping orders.

SALE POLICY

We guarantee that all our products work in an A.C.E. configuration microcomputer. We will endeavour to assist in custom applications, but assume no liability for such use. Orders will be shipped as promptly as payment is guaranteed.

NAME: ______________________________

MAILING ADDRESS: ______________________________

____

____

____

PHONE NO.: ______________________________

Note: Ensure mailing address is correct, complete and printed.
Please ensure payment is enclosed.

ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS
c/o M.E. FRANKLIN
690 LAURIER AVENUE,
MILTON, ONTARIO
L9T 4R5
