Ipso Facto

ISSUE 36 AUGUST, 1983

INDEX PAGE

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

Executive Corner	2
Editor's Corner	3
Members' Corner	4
More on Pittman's DOTS for the 1802	5
64 Character Line for Tiny Pilot	6
ACE Tiny Pilot Review	7
A Simple 2716 EPROM Programmer	9
Line Generator for the 6847 High Resolution	14
The WD 2412 Time of Day Clock	23
Centronics Parallel Port for the Super ELF	30
Electronic Power Supply Load Tester	32
Notes on Don Stevens Notable Assembler	33
A Simple Controller	34
CHIP 8 Games Mods for the 6847	36
ACE Conference	39
Club Communique and Membership Renewal Form	41

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.

Editor's Corner

This issue represents the end of Year 6 for ACE. It has been a long, and sometimes difficult year as Editor, with lack of articles or member input plaguing the Newsletter output. It may be fashionable to blame the economy for decreased interest, or a surplus of "high tech" articles which are of little interest to "low tech" oriented members, but I think it is more serious than that.

The ACE system costs approximately $750.00 to get 64K, video (32 x 16), 8" single sided disk, no case, and little software, etc., compared to similarly priced Vic 64's, Atari's, TRS-80's, Epsoms, etc., with a myriad of software packages from which to choose. The Netronics ELF II, Quest Super Elf and Venture are similarly priced when "blown up" to a complete system, and again with limited software. These 1802 systems are still in the low-end of the micro-computer price spectrum of the market; still about the only "kit" learning system; and still the cheapest to get started - about $120.00 with a 1/4k of RAM and a "Q" LED. If you think back to the reasons why you bought, or built an ELF (kit building, low cost and popularization of the CMOS 1802 by Popular Electronics probably had a lot to do with it), and compare it to your aspirations or needs of today - you will probably recognize a distinct gap in performance - particularly software.

Our old trusty CMOS 1802 from a hardware point of view is capable of anything a Z80 or 6502 can do (perhaps a little slower), but its operating software has decidedly lagged behind the other micro chips.

ACE, through its meetings, the Newsletter and the conferences, provides a forum for disemminating current information, programs and circuitry to its 500 members world-wide. It remains about the only source of un-commercialized 1802 information.

ACE started out as an experimenter's club for ex-kit builders and "make it yourself" ham radio operators. The expensive plastic boxes and someone else's software were for the consumerists. For me, ACE is still that way, and for those with similar interests, ACE will still be here next year - to welcome you back. For those of you who want BASIC, pap or pre-assembled and pre-tested boards in plastic cases - well you're still welcome to join, but don't expect that kind of product from a group of guys with micros mounted on hinges on a 2 x 4 stud wall in the basement, watching programs on gutted TV's and jiggling wires on a wire-wrapped proto-type to see what happens. We just don't think that way.

To us, and we hope to you, there is more satisfaction in getting a new board up and running, or writing our own monitors, DOS, or customized home security software than in unwrapping a factory-assembled box and plugging in someone else's software. Each little victory is worth sharing with others so they may learn from our experience; each little problem worthy of debate or questioning of our peers.

I would encourage you to seriously think about what you want from a micro-computer and from a club before you renew next year. If you can relate to what I have said, then get our your cheque/check book and pen; if you can't, then I hope you can find what you want from some other micro manufacturer and from some other club.

Hope to see you at the conference in August.

Happy computing......

Members' Corner

FOR SALE:

George Musser, 60 Broadway Road, Warren, NJ 07060 (201) 647-1437.

Complete Netronics ELF II System. Includes:

ELF II motherboard	ACE VDU Board
Giant Board	28K RAM
ASC II keyboard	Epson printer interface board
ACE Netronics Adapter Board	Complete enclosure & power supply

Extensive software, literature, periodicals & documentation.
Worth over $1,150.00

David Schuler, 3032 Avon Road, Bethlehem, PA. 18017 (215)865-1188

Two SSM (Solid State Music) MB6B 8K S-100 static RAM boards with all chips and documentation, compatible with Quest S-100 expansion and other S-100 bus adapters in Ipso Facto. $55.00 each or both for $100.00.

Four INTEL 8086 16 bit microprocessors - $25.00 each or 4 for $80.00.

Western Digital FD1973B-02 Double Density Floppy Disk Controller - $35.00.

Five precision Monolithics DAC888 microprocessor compatible 8 bit D/A converters - $2.00 each or 5 for $8.00.

I am willing to negotiate any of the above prices in trade for 6116 2K x 8 200 ns static RAM's.

Mike Franklin, 690 Laurier Avenue, Milton, Ontario. L9T 4R5

Traded up to complete ACE system - wish to sell trusty ELF II.

1 - ELF II, modified for NAB and ACE mini boot;
1 ea. NAB, Giant Board, 16K static RAM, kluge and math board (converted to 4-2K EPROM).
1 - Netronics keyboard.

All in working order, with full documentation. $150.00 complete.

1 - 8" disk drive, controller board, cables, etc.
Working Order - $200.00 complete.

HELP!

J.H. Golbeck, 4876 Melbourne Rd., Baltimore, Maryland. 21229

I have an ELF II with Giant Board, 16K of memory, 16K of EPROM and an ASR 33 for I/O. I am interested in obtaining a version of FORTH that includes an editor and assembler than can run either as is or with a minimum number of changes on my system. If anyone has patched in the editor and assembler and would be willing to either share or sell a cassette tape of this package, please contact me at the address above.

More on Pitman's Dots for the 1861

- by C.C. Goodson, Av. Francisco Glicerio 467, #501, Campinas, S.P. Brazil

As far as I know, I am the only 1802 experimenter in Brazil, and the discovery of ACE and Ipso Facto has been a real help and encouragement. Perhaps some others would be interested in my problems and solutions with Tiny Basic on ROM. Mr. Swindells article in IF29, page 15, received my full attention. I have Quest Tiny Basic V1.1 in ROM at 8400H and the Quest Super Monitor ROM V2.0 at 8300H. This requires the user to supply his own I/O routines; in my case, a parallel keyboard and an 1861 display.

Mr. Pittman's article, "DOTS" (Kilobaud, April 1979, page 34) seemed the solution for my driver. After keying it in and making a cassette copy, I ran it with the Test Main routine, and was rewarded with the flashing cursor on the screen. After correcting for the proper parallel input port, I had a TV typewriter. So, I punched up the Super Monitor option for a Basic cold start, and like Mr. Swindells, was rewarded with the program crashing. The DOTS program needed to be moved to a location where it did not conflict with the I/O jumps of the Tiny Basic ROM at 0100-0108. Tiny Basic also had to be told not to use the memory occupied by DOTS. How this was to be done, was not clear, but a letter to Mr. Pittman received a prompt and helpful explanation: On a cold start, Tiny does a non-destructive examination of memory and records the limits at 0020-23H. I only had to do a cold start with nothing at the jump points 0100-0108, reset, and alter the contents at 0020-23 to the desired memory limits. After that, use only warm starts.

Moving DOTS was a real exercise in analyzing the function of a program. The program was moved from 00A0-o44E to 01A0-054E, to clear the jump points at 0100-0108. The block move of memory is just a matter of a few key pushes with the Quest Super Monitor, but the program changes to compensate for the move was another matter. Jumps to memory addresses below 00A0 must be left as they are to agree with Tiny Basic, but the following addresses must have their contents increased by 1 to compensate for the new page numbers: (Old addresses, as in Kilobaud)

00A4, 00B5, 00B8, 00BE, 01C2, 01CE, 01F3, 0252, 025C, 0277, 0282, 028F, 029C, 02AA, 02EE, 06FA
0359, 0364, 036D, 03CE, 03DD

Also, the following changes were made to compensate for the differences between the Super Elf hardware and the Elf II: 00BA to 3D, 01C0 to 6D, 01DA to 62. But when 01A8 was changed to command the TV ON, the program would crash, so it was left unchanged. Perhaps someone can explain this quirk to me?

As progress was made in finding and making these changes, I was able to write a T-Basic command, such as PRINT 3-2. Then the shortcomings of my flea-market keyboard became really apparent. In spite of having a bunch of special keys, it did not have: ENTER, +, (,), Formfeed, ", ?, =, lowercase, and other essential signals! By placing switches in 2 of the output lines, I was able to supply these, if somewhat awkwardly. Now the PRINT 3-2 gave a 1 on the next line with a new prompt below. Progress!

But when I tried to enter and RUN a program, TBasic gave me an error code indicating it had nothing in memory to run! A LIST gave me the same result. My programs were not being remembered! Hours of single-stepping and study finally revealed the over-looked byte change, so that I could write a program and LIST it. (Change included in list above). However, to my dismay, when I typed RUN, it executed only the first line, and then accused me of ordering a BREAK before the second! A GOTO (2nd line) gave the same accusation. I could only write programs of one line. At 0106 I had merely placed a D5 instruction to jump back to TBasic without looking for a break, but it obviously jumped back already loaded for a break. Mr. Swindells' article touched on this difficulty, but his solution was hidden in his monitor routine, which I did not have. A Tiny Basic to TTY Interface program in Questdata V.I No. 12 page 3, included a break test routine, from which FC 00 D5 was loaded at 0106 and Tiny was up and running!

I envied Mr. Swindells' opportunity to consult directly with Mr. Bevis and Mr. Murphy, but perhaps I would have gained a lot less experience if such help had been at hand. Even so, it is a real boost to the morale to know that there is help available by letter when I really am stuck!

64 Character Line for Tiny Pilot (Ipso Facto #33)

- by John Deering, 15709 Fresno Street, Victorville,Ca. 92392

Hooray for Wayne Bowdish and Tony Hill for a nice revision of R.W. Petty's Tiny Pilot. Just punch it in and use it. But if you have a Super Elf with a VB1B video board like me and want a full screen with a 64 character line, there is more work to do. The output routine is there in Tiny Pilot at 0770H - so use it. I used the Disassembler by Wayne Bowdish to see what the output routine was, and punched in changes until the display was right. This is what I came up with:

For my Super Elf, the keyboard input code was changed at address 0769 to 3D 69 6D. In the output routine I made the following changes:

```
At 0790 FA 7F        STRIP PARITY BIT (A TYPO, I BELIEVE)
   07AA 04           # OF PAGES DISPLAYED (1K SCREEN)
   07CA, 07EB 40     LINE LENGTH +1 (64 CHARACTERS)
   07C6 C0           STARTS NEXT LINE AT THE LEFT SIDE OF PAGE
   0796, 07B6, 07D2, 07D6, 07E3, 07F5 E4 LAST SCREEN PAGE +1
```

To return to system monitor using the jump at address 0013 (C0 10 00) I had to place the following code at address 1000:

```
1000 F8 AA        LOAD ANY NON-ZERO NUMBER
1002 F8 80 B0     LOAD ENABLE MONITOR
1005 F8 54 A0     INTO REGISTER 0
1008 D0           MAKE REGISTER 0 THE PROGRAM COUNTER
```

After researching Petty's Pilot and Cuddly Software's Pilot I/O Program, I found:

TOPMEM: .EQL #20 Highest RAM Page available for text buffer to be located at addr. 027A and 02BA. You can change this to fit your system.

NEXT, have fun with TINY PILOT.

ACE Tiny Pilot Review

- R. Carr, 4691 Freeman Road, Middleport, N.Y. 14105

After reviewing the 1802 Tiny Pilot in the February '83 issue of Ipso Facto, I thought a short article comparing the Netronics Tiny to the ACE version would be in order, especially since much was left UNSAID in the Ipso article about using Tiny Pilot.

First, for anyone bringing up Tiny Pilot for the first time, I would suggest they practice using the editor and move the pointer around until they are familiar with its operation. The pointer is always set to the start of a line, and text will be entered (inserted) before this point, i.e., before the pointer and after the preceding line. All text input to Tiny Pilot must begin with the Insert Command "I",

i.e.,

```
IMY NAME IS BOB
IK: 12
```

When you use the Write Command "W" to list back any entered text, the "I" will not be printed. You must position the text pointer before using the "W" command. Do this with the "B,E,U and D" commands. All Editor Commands are entered as the first character on a line,

i.e.,

```
B
K 12
```

Notice that the Editor and Tiny Pilot share some syntax commands which have DIFFERENT meanings. In the first example, IK:12 executes within Tiny by outputting a form feed to your terminal. In the second case, K12 KILLS 12 lines of text - so be careful. A comparison of all Editor and Pilot commands is at the end of this article.

NOW for some of the differences. ACE PILOT uses a Replace Command for changing text on the current line. This is similar to the Netronics CTL X function which erases the current line, allowing it to be re-entered. Netronics also has three commands with no similar ACE counterpart; "CTL I" which invokes a H TAB (4 spaces), and "Width" which is used to set the line width to your terminal, and "&" for placing comments on a Tiny Pilot line.

The ACE Pilot has a couple of unique commands of its own. First and foremost is the SCRT command. This is a very useful addition to a limited language, and the very best reason to make ACE Pilot the Tiny Pilot of your choice. The other useful command is the "high" which allows you to find the end of the text buffer so you can use your monitor to save and load a program. Netronics has a Length Command but it shows 1 + the last 256 byte page of available memory.

A quick look at the long branches at the start of ACE Tiny Pilot pointed me to the end of the listing, which contains a driver for the ACE VDU. This means that the area from 0769H to 07FFH is available for SCRT subroutines for those with other types of drivers.

Comparison of ACE Tiny Pilot and Netronics Tiny Pilot Commands

ACE	EDITOR INSTRUCTION	NETRONICS	ACE	PILOT INSTRUCTION	NETRONICS
B	BEGIN	B	A	ASK	A
C	CLEAR	C	C	COMPUTE	C
D	DOWN	D	E	END	E
E	END	E	J	JUMP	J
H	HIGH	L	K	CONTROL	K
I	INSERT	I	M	MATCH	M
K	KILL	K	N	NO	N
L	LOAD	A	R	RETURN	R
M	MONITOR	M	S	SCRT CALL	-
P	RUN PILOT PROGRAM	R	T	TYPE	P
R	REPLACE	CTL X	U	USE	S
S	STORE	S	X	EXAMINE	X
U	UP	U	Y	YES	Y
W	WRITE	T	Z	RANDOM	Z
CTL H	BACKSPACE	CTL H	-	REMARKS	&
	WIDTH	W	%	LINE	%
	TAB	CTL I			

A Simple 2716 EPROM Programmer

- by D. Schuler, 3032 Avon Rd., Bethlehem, PA. 18017

The ability of EPROM's to retain data when power is removed and subsequently restored makes them ideal for storing system bootstrap and monitor routines that are used each time the system is started. After using 2708 EPROM's for the past year and one-half, I decided that it was time to upgrade to a memory board that used the new 6116 (2KX8) RAM/2716 EPROM devices interchangeably on the same board.

Up to this time I was using the EPROM programmer that had been featured in Popular Electronics; and did not want to scrap it for a 2716 programmer. After reviewing the 2716 EPROM programming requirements against the 2708 requirements, I decided that this programmer could be utilized with only a few minor changes to my existing hardware and software.

HARDWARE MODIFICATIONS

The 2716 differs from the 2708 in the following areas:

- -11 address bits are required for the 2716 vs. 10 for the 2708
- -the 2716 requires only a +5 volt supply vs. +5, +12, and -5 for the 2708 (both require +25 for programming)
- -the 2716 requires only one 50msec pulse at each location to be programmed, and any location can be programmed independently of all others vs. the 2708 requiring appx. 100 loops of 2msec each in constantly ascending order

The schematic for the EPROM programmer in Figures 1 and 2 is based upon the programmer in the Popular Electronics article, but has a few major changes. (For a complete description of the theory of operation, see the article in the June 1981 issue of P.E.) Overall, the circuit has been simplified to eliminate all unnecessary devices, and standard 74XX series devices have been specified in most cases. Depending on your system and budget requirements, these devices can be either 74LSXX or 74CXX for either low power or lowest power requirements respectively. Note that the +12 and -5 volt supplies are no longer required for the 2716 devices, and therefore can be eliminated from existing programmers. Be very careful that ALL necessary changes have been performed before using the programmer if you are converting from the 2708 version, as the voltages are different and can "fry" a 2716 very quickly.

SOFTWARE

Listing 1 contains a new version of the programming software for use with the 2716 EPROM programmer. This routine will first check to see if the EPROM has been completely erased bu the UV light source. If this is done, the software will then ask for the RAM starting address of the data to be placed in EPROM. (GETADD is a user-supplied routine.) After this, the EPROM will then be programmed under complete control of the software. The total programming time for all 2048 bytes is 102±1 seconds. After the EPROM has been programmed, its contents are then checked with that of the source RAM, and the result of this test will be displayed. This entire program is designed to be called using the SCRT call/return technique. It can be incorporated into an existing system monitor, or can be designed to run independently with the appropriate supporting routines.

After installing the software and BEFORE attempting to program an EPROM, the software routine should be tested to verify that the delay constant is correct. A "dry run" should be performed, and the time required to execute the routine should be measured with a stopwatch. It should take a total of 103±1 seconds to run completely. If the running time is outside of this range, adjust the delay constant (increase if less than 103 seconds and decrease if over 103 seconds) to bring it to within acceptable limits.

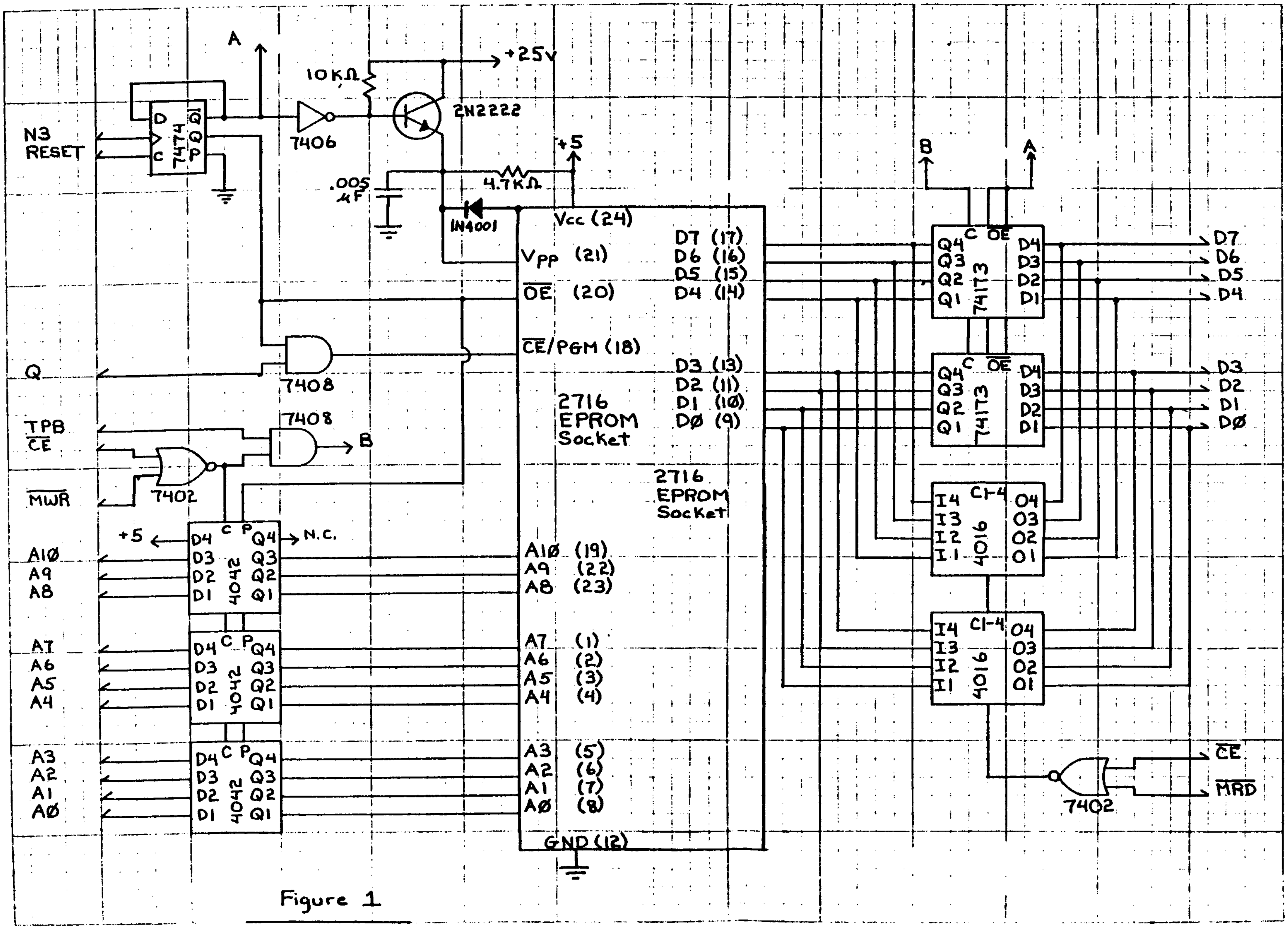
The preceeding circuit and software enable a 2716 EPROM to be programmed under software control in less than 2 minutes. In the future as higher density devices (i.e. 2732, 2764, and 27128) become affordable, this circuit can again be modified to take advantage of these increased bit densities.

REFERENCES:

Bregoli, Larry, "An 1802-Based EPROM Programmer", Popular Electronics, June 1981.

"Am2716/Am9716 Product Specification", Advanced Micro Devices, April 1981.

"Archer Technical Data - 2716 EPROM", Radio Shack.



A
+25V
10KΩ
2N2222
7474
7406
N3
RESET
D
Q
C
P
+5
.005 µF
4.7KΩ
1N4001
Vcc (24)
VPP (21)
OE (20)
CE/PGM (18)
D7 (17)
D6 (16)
D5 (15)
D4 (14)
D3 (13)
D2 (11)
D1 (10)
DØ (9)
2716 EPROM Socket
A1Ø (19)
A9 (22)
A8 (23)
A7 (1)
A6 (2)
A5 (3)
A4 (4)
A3 (5)
A2 (6)
A1 (7)
AØ (8)
GND (12)
7408
TPB
CE
B
7402
MWR
4042
N.C.
74173
4016
MRD

Figure 1

```
..PROGRAM 2716 TYPE EPROM
..USE OF SCRT CALL / RETURN TECHNIQUES IS ASSUMED
EPROM:  CALL OSTRNG ,T'PROGRAM 2716 EPROM' ,#ØDØ3  ..PRINT SIGN-ON
        LDI #DØ; PHI RD  ..LOCATION OF EPROM PROGRAMMER IN MEMORY
        LDI #ØØ; PLO RD
        LDI #Ø8; PHI R9 ..LENGTH OF 2716 EPROM + #Ø1ØØ
        LDI #FF; PLO R9
ERASE:  LDA RD; XRI #FF; BNZ NOGO  ..CHECK IF EPROM IS ERASED?
        GHI R9; BZ OK
        DEC R9; BR ERASE
NOGO:   CALL OSTRNG ,T'? ERASE' ,#ØDØ3  ..EPROM NOT ERASED
        RETURN  ..RETURN TO CALLING PROGRAM
OK:     CALL GETADD  ..GET STARTING ADDRESS OF DATA TO COPY, RETURN
                     ..IN RC (USER SUPPLIED ROUTINE)
        GHI RC; PHI RB  ..COPY RC TO RB
        GLO RC; PLO RB
        LDI #DØ; PHI RD  ..LOCATION OF EPROM PROGRAMMER IN MEMORY
        LDI #ØØ; PLO RD
        LDI #Ø8; PHI R9  ..LENGTH OF 2716 EPROM + #Ø1ØØ
        LDI #FF; PLO R9
        OUT 3; DEC STACK  ..TURN ON PROGRAMMING VOLTAGE
        ..BEGIN PROGRAMMING EPROM
NXTBTE: LDN RB; STR RD  ..LATCH DATA INTO PROGRAMMER
        DEC R9; INC RB; INC RD  ..ADJUST POINTERS
        ..BEGIN 50msec DELAY  (MUST BE NO GREATER THAN 55msec)
        LDI #Ø6; PHI RE  ..DELAY COUNT  (MAY NEED TO BE ADJUSTED
        LDI #ØØ; PLO RE  ..IN SOME SYSTEMS)
        SEQ
DELAY:  DEC RE
        GHI RE; BNZ DELAY  ..DELAY LOOP
        REQ
        GHI R9; BNZ NXTBTE  ..REPEAT UNTIL ALL BYTES PROGRAMMED
        OUT 3; DEC STACK  ..TURN OFF PROGRAMMING VOLTAGE
```

<u>LISTING 1</u>

```
        ..VERIFY CONTENTS OF EPROM AGAINST RAM
        LDI #Ø8; PHI R9   ..LENGTH OF 2716 EPROM + #Ø1ØØ
        LDI #FF; PLO R9
        LDI #DØ; PHI RD   ..LOCATION OF EPROM PROGRAMMER IN MEMORY
        LDI #ØØ; PLO RD
        SEX RC
VERIFY: LDN RD; XOR; BNZ ERROR  ..COMPARE EPROM AND RAM
        INC RD; IRX; DEC R9  ..ADJUST POINTERS
        GHI R9; BNZ VERIFY  ..REPEAT UNTIL ALL BYTES CHECKED
        CALL OSTRNG ,T'OK' ,#ØDØ3  ..EPROM PROGRAMMED CORRECTLY
        RETURN  ..RETURN TO CALLING PROGRAM
ERROR:  CALL OSTRNG ,T'PROGRAMMING ERROR - DOES NOT VERIFY' ,#ØDØ3
        RETURN  ..RETURN TO CALLING PROGRAM
        ..
        END
```

LISTING 1

Line Generator for the 6847 High Resolution

- by Lynn Keenliside, London, Ontario. B.C.

It took me a long time, but I finally made a contribution. This week I learned how to use an assembler and it leaves me without an excuse to delay any longer.

The program is the beginning of a graphics terminal like the Tektronix. Given a beginning X,Y, in register D, and an ending X,Y in register E, it will draw a line between them. The main body fits in one page with subroutines on a second page.

The calling sequence would be D4 OB 00 OD to set up a table on page OD00. Next D4 OB 02 OD will draw the line using the table on page OD00 and RD/RE for X,Y points. Registers other than S and F are restored. (These are used as scratch and are not maintained). In the table X and Y offsets permit moving the origin from bottom left to any point desired. VStart can be changed to new start, or leave a window for text. The write flag, if not zero, will erase a line rather than draw it. The bit map corresponds to the 8 bit positios available in graphics six R mode.

One other point I should mention - my video screen starts at E800 (bottom right) ends at FFFF (top left). This is due to inverting address MUX's. It seemed logical that top of screen should be top of RAM.

```
                                ;
                                ; THIS PGM CREATES A LINE
                                ; BETWEEN 2 GIVEN X,Y POINTS.
                                ;FOR 6847 VIDEO CONTROLLER
                                ;
                                ;WRITEN MAY 1983 BY L. KEENLISIDE
                                ;
                                ;RD & RE HAVE X1,Y1 & X2,Y2  *NOT ALTERED
                                ;RA, RB, & RC ARE SAVED & RESTORED.
                                ;R8 IS USED FOR SCRATCH, X & Y INC
                                ;RA IS VIDEO MEMORY ADDRESS
                                ;R.B IS CURRENT BIT POSITION
                                ;RB. IS TEMPORARY WRITE FLAG
                                ;RC IS TABLE POINTER
                                ;RF IS SCRATCH, DELTA X & SIGN
                                ;
      0B00                      .= $0B00        ;START ADDRESS
                                     ;
0B00  30   E2         SETUP:: BR MAKTA          ;THIS IS VECTOR FOR SETUP
0B02  E2              LINE::  SEX R2
0B03  97                      GHI R7            ;SAVE MACRO POINTER
0B04  73                      STXD
0B05  F8   0C                 LDI $0C           ;TEMPORARY MACRO ROUTINES
0B07  B7                      PHI R7
0B08  F8   02                 LDI $02           ;AT 0C02
0B0A  A7                      PLO R7            ;SET LOW JUST TO BE SURE
0B0B  D7                      SEP R7            ;CALL MACRO
0B0C  0A                      .BYTE SAVAE       ;SAVE REGISTERS A TO E
0B0D  46                      LDA R6            ;GET TABLE PAGE
0B0E  BC                      PHI R12
0B0F  D7                      SEP R7            ;CONVERT X2,Y2 TO ABSOLUTE
0B10  14                      .BYTE PNTAD       ;SUB#14, 2 STACK BYTES LOST
                                                ;WE NOW HAVE ABSOLUTE POINT OF X2,Y2
                                                ;IN RA, BIT MASK IN R.B
0B11  F8   1F                 LDI MASK                ;POINT TO TABLE
0B13  AC                      PLO R12           ;THIS WILL BE
0B14  EC                      SEX R12           ;LAST POINT OF LINE
0B15  8B                      GLO R11           ;
0B16  73                      STXD              ;SAVE BIT MASK
0B17  9A                      GHI R10           ;
0B18  73                      STXD              ;SAVE LOW ADDRESS
0B19  8A                      GLO R10           ;
0B1A  73                      STXD              ;SAVE HI  ADDRESS
0B1B  E2                      SEX R2
0B1C  D7                      SEP R7            ;GET ABSOLUTE OF X1,Y1 ON STACK
0B1D  14                      .BYTE PNTAD       ;SUB#14 2 STACK BYTES DROPPED
0B1E  F8   20                 LDI $20           ;SET YINCR TO 32
0B20  A8                      PLO R8
0B21  F8   01                 LDI $01           ;SET XINCR TO 1
0B23  B8                      PHI R8
0B24  E2                      SEX R2
0B25  F8   17                 LDI DELTX         ;POINT TO DELTA X
0B27  AC                      PLQ R12
0B28  9D                      GHI R13
```

```
0B29  52              STR R2
0B2A  9E              GHI R14
0B2B  F7              SM              ;CALCULATE DELTA X
0B2C  5C              STR R12         ;AND SAVE IT
0B2D  AF              PLO R15         ;IN 2 PLACES
0B2E  1C              INC R12
0B2F  F8  00          LDI $00
0B31  7E              SHLC
0B32  5C              STR R12         ;SAVE SIGN FLAG
0B33  BF              PHI R15         ;IN 2 PLACES
0B34  1C              INC R12         ;POINT TO DELTA Y
0B35  8D              GLO R13
0B36  52              STR R2
0B37  8E              GLO R14
0B38  F7              SM
0B39  5C              STR R12         ;SAVE DELTA Y
0B3A  1C              INC R12
0B3B  F8  00          LDI $00
0B3D  7E              SHLC
0B3E  5C              STR R12         ;SAVE SIGN FLAG
0B3F  1C              INC R12
0B40  1C              INC R12         ;POINT TO ERROR TERM SIGN
0B41  EC              SEX R12
0B42  F8  00          LDI $00         ;CLEAR ERROR TERM
0B44  73              STXD
0B45  5C              STR R12
0B46  F8  15          LDI WRTFLG      ;MOVE FLAG
0B48  AC              PLO R12
0B49  0C              LDN R12
0B4A  B8              PHI R11         ;TO KEEP IT HANDY
0B4B  9F              GHI R15         ;GET DELTA X SIGN
0B4C  3A  58          BNZ TSTZX       ;BR IF SIGN PLUS
0B4E  F8  18          LDI DELTX+1     ;POINT TO DELTA X SIGN
0B50  AC              PLO R12
0B51  D7              SEP R7          ;CALL MACRO
0B52  16              .BYTE NEG2      ;NEGATE DELTA X
0B53  AF              PLO R15         ;SAVE DELTA X
0B54  98              GHI R8          ;GET XINCREMENT
0B55  FD  00          SDI $00         ;NEGATE XINCREMENT
0B57  B8              PHI R8          ;SAVE XINC
0B58  8F      TSTZX:  GLO R15         ;GET DELTA X
0B59  3A  62          BNZ TSTY        ;IF DELTA X=0 E= -1
0B5B  F8  1C          LDI ERTRM+1     ;POINT TO ERROR SIGN
0B5D  AC              PLO R12
0B5E  F8  FF          LDI $FF
0B60  73              STXD
0B61  5C              STR R12         ;SET ERROR TERM -1
0B62  F8  1A  TSTY:   LDI DELTY+1     ;POINT TO DELTA Y SIGN
0B64  AC              PLO R12
0B65  0C              LDN R12
0B66  3A  7D          BNZ PLOT
0B68  D7      NEGY:   SEP R7          ;CALL MACRO
0B69  16              .BYTE NEG2      ;NEGATE DELTA Y
0B6A  88              GLO R8          ;GET YINCREMENT
```

```
0B6B  FD  00             SDI $00        ;NEGATE IT
0B6D  A8                 PLO R8         ;SAVE YINC
0B6E  30  7D             BR PLOT        ;READY TO PLOT LINE
                                        ;
                                        ;
0B70  AB          XE:    PLO R11        ;UPDATE MASK
0B71  F8  19             LDI DELTY      ;E= E - DELTA Y
0B73  AC                 PLO R12        ;POINT TO DELTA Y
0B74  4C                 LDA R12
0B75  1C                 INC R12        ;POINT TO ERROR TERM LOW
0B76  F5                 SD             ;SUBTRACT DELTA Y
0B77  5C                 STR R12        ;FROM ERROR TERM
0B78  1C                 INC R12
0B79  0C                 LDN R12
0B7A  7F  00             SDBI $00       ;SUBTRACT CARRY
0B7C  5C                 STR R12
0B7D  9A          PLOT:  GHI R10        ;TEST OUT OF BOUNDS
0B7E  FF  E8             SMI $E8        ;E800 TO FFFF OK
0B80  3B  8F             BNF PDOT+1
0B82  9B                 GHI R11        ;GET WRITE FLAG
0B83  EA                 SEX R10
0B84  32  8A             BZ WRITE       ;BR WRITE IF SET
0B86  8B                 GLO R11        ;GET BIT MASK
0B87  F1                 OR             ;REMOVE DOT
0B88  30  8E             BR PDOT
0B8A  8B          WRITE: GLO R11        ;GET BIT MASK
0B8B  FB  FF             XRI $FF        ;COMPLIMENT ALL
0B8D  F2                 AND            ;ADD DOTS
0B8E  5A          PDOT:  STR R10        ;REPLACE DOTS ON SCREEN
0B8F  F8  1D             LDI LAST       ;POINT TO LAST POINT ADDRESS
0B91  AC                 PLO R12
0B92  EC                 SEX R12
0B93  8A                 GLO R10        ;TEST FOR LINE END
0B94  F3                 XOR            ;LAST LINE YET?
0B95  3A  A8             BNZ NOTFIN     ;BR NOT FINISHED
0B97  1C                 INC R12
0B98  9A                 GHI R10
0B99  F3                 XOR            ;END YET?
0B9A  3A  A8             BNZ NOTFIN     ;BR NOT
0B9C  1C                 INC R12
0B9D  8B                 GLO R11
0B9E  F3                 XOR            ;LAST BIT YET?
0B9F  3A  A8             BNZ NOTFIN     ;BR NOT
0BA1  E2                 SEX R2
0BA2  D7                 SEP R7         ;CALL MACRO
0BA3  1C                 .BYTE RSTCA    ;RESTORE REGISTERS C TO A
0BA4  60                 IRX            ;RD & RE HADN'T CHANGED
0BA5  F0                 LDX            ;RESTORE OLD MACRO POINTER
0BA6  B7                 PHI R7
0BA7  D5                 SEP R5         ;RETURN ALL DONE
0BA8  F8  1C      NOTFIN: LDI ERTRM +1  ;POINT TO ERROR TERM SIGN
0BAA  AC                 PLO R12
0BAB  0C                 LDN R12
0BAC  FE                 SHL            ;IS ERROR NEGATIVE?
```

```
0BAD  3B  CF              BNF UPX       ;IF PLUS MOVE X DIRECTION
0BAF  2C          ADDX:   DEC R12       ;E= E+ DELTA X
0BB0  8F                  GLO R15       ;GET DELTA X
0BB1  F4                  ADD           ;ADD DELTA X TO ERROR
0BB2  5C                  STR R12
0BB3  1C                  INC R12
0BB4  0C                  LDN R12
0BB5  7C  00              ADCI $00      ;ADD CARRY
0BB7  5C                  STR R12
0BB8  88                  GLO R8        ;GET YINCREMENT
0BB9  E2                  SEX R2
0BBA  52                  STR R2        ;ONTO STACK FOR MATH
0BBB  FE                  SHL           ;MOVE IN Y DIRECTION
0BBC  8A                  GLO R10
0BBD  3B  C7              BNF ABCD
0BBF  F4                  ADD           ;MOVE IT DOWN
0BC0  AA                  PLO R10
0BC1  9A                  GHI R10
0BC2  7C  FF              ADCI $FF
0BC4  BA                  PHI R10
0BC5  30  70              BR PLOT
0BC7  F4          ABCD:   ADD           ;MOVE IT UP
0BC8  AA                  PLO R10
0BC9  9A                  GHI R10
0BCA  7C  00              ADCI $00
0BCC  BA                  PHI R10
0BCD  30  70              BR PLOT
0BCF  98          UPX:    GHI R8        ;GET XINCREMENT
0BD0  FE                  SHL           ;DO WE SHIFT L OR R
0BD1  8B                  GLO R11       ;GET BIT MASK
0BD2  3B  DB              BNF XRT
0BD4  FE                  SHL
0BD5  3A  70              BNZ XE
0BD7  7E                  SHLC
0BD8  1A                  INC R10
0BD9  30  70              BR XE
0BDB  F6          XRT:    SHR
0BDC  3A  70              BNZ XE
0BDE  76                  SHRC
0BDF  2A                  DEC R10
0BE0  30  70              BR XE
                          ;
                          ;THAT'S IT
                          ;
0BE2  46          MAKTA:  LDA R6        ;GET TABLE PAGE
0BE3  B8                  PHI R8
0BE4  F8  16              LDI $16       ;TOP OF TABLE
0BE6  A8                  PLO R8
0BE7  E8                  SEX R8
0BE8  F8  00      CLR:    LDI $00
0BEA  C8                  LSKP
0BEB  F8  E8      VST:    LDI $E8       ;VSTART
0BED  73                  STXD
0BEE  88                  GLO R8
```

```
0BEF  FB  14              XRI $14       ;TEST FOR VSTART POSITION
0BF1  32  EB              BZ VST
0BF3  88                  GLO R8
0BF4  3A  E8              BNZ CLR
0BF6  F8  80              LDI $80       ;LEFT MOST BIT
0BF8  58          MAP:    STR R8
0BF9  18                  INC R8
0BFA  F6                  SHR           ;NEXT BIT
0BFB  3A  F8              BNZ MAP
0BFD  D5                  SEP R5        ;RETURN

                          ;
                          ;
                          ;MACRO SUBROUTINES
                          ;FOR SYSTEM MONITOR ETC.
                          ;
                          ;CALLING SEQUENCE IS-
                          ;SEP R7  .BYTE
                          ;WHERE .BYTE IS THE SUB #
                          ;THIS MAY BE FOLLOWED BY
                          ;ANY INLINE BYTES REQUIRED
                          ;
                          ;
    E100                  .= $E100      ;START
                          ;
E100  9F          FINI:   GHI R15       ;RETURN WITH D AS BEFORE
E101  D3                  SEP R3        ;RETURN TO CALLER
E102  BF          MACRO:: PHI R15       ;KEEP D FOR RETURN
E103  43                  LDA R3        ;GET SUB #
E104  A7                  PLO R7        ;BR TO VECTOR
E105  C0  E1  00  MTABLE::LBR FINI      ;DUMMY
E108  30  20      STR2X:  BR VSTR2X     ;STORE 2 INLINE VIA REG X
E10A  30  26      SAVAE:: BR VSAVAE     ;SAVE REG A TO REG E VIA X
E10C  30  00              BR FINI       ;DUMMY
E10E  30  74      TSTDE:  BR VTSTDE     ;TEST RD>RE
E110  30  00              BR FINI       ;DUMMY
E112  30  54      ASCI:   BR VASCI      ;CONVERT ASCII TO NYBLE
E114  30  80      PNTAD:  BR VPNTAD     ;POINT ADDRES OF X,Y
E116  30  B5      NEG2:   BR VNEG2      ;NEGATE 2 BYTES VIA RC
E118  30  B9      NEG1:   BR VNEG1      ;NEGATE BYTE VIA RC
E11A  30  3C      RSTEA:  BR VRSTEA     ;RESTORE REG'S E TO A
E11C  30  45      RSTCA:  BR VRSTCA     ;RESTORE REG'S C TO A
E11E  30  C1      TOP:    BR VTOP       ;TEST D > BF
    E120                  .= $E120      ;BEGIN MACROS
E120  43          VSTR2X: LDA R3        ;SAVE 2 ILINE BYTES VIA RX
E121  73                  STXD
E122  43                  LDA R3
E123  73                  STXD
E124  30  00              BR FINI
E126  9A          VSAVAE: GHI R10
E127  73                  STXD
E128  8A                  GLO R10
E129  73                  STXD
E12A  9B                  GHI R11
```

```
E12B  73                STXD
E12C  8B                GLO R11
E12D  73                STXD
E12E  9C                GHI R12
E12F  73                STXD
E130  8C                GLO R12
E131  73                STXD
E132  9D                GHI R13
E133  73                STXD
E134  8D                GLO R13
E135  73                STXD
E136  9E                GHI R14
E137  73                STXD
E138  8E                GLO R14
E139  73                STXD
E13A  30   00           BR FINI
E13C  60        VRSTEA: IRX
E13D  72                LDXA
E13E  AE                PLO R14
E13F  72                LDXA
E140  BE                PHI R14
E141  72                LDXA
E142  AD                PLO R13
E143  F0                LDX
E144  BD                PHI R13
E145  60        VRSTCA: IRX            ;MOVE UP TO DATA
E146  72                LDXA
E147  AC                PLO R12
E148  72                LDXA
E149  BC                PHI R12
E14A  72                LDXA
E14B  AB                PLO R11
E14C  72                LDXA
E14D  BB                PHI R11
E14E  72                LDXA
E14F  AA                PLO R10
E150  F0                LDX
E151  BA                PHI R10
E152  30   00           BR FINI
E154  9F        VASCI:  GHI R15
E155  FF   30           SMI $30
E157  3B   6F           BNF NOASC
E159  FF   0A           SMI $0A
E15B  3B   69           BNF LETR
E15D  FF   07           SMI $07
E15F  3B   6F           BNF NOASC
E161  FF   06           SMI $06
E163  33   6F           BDF NOASC
E165  9F                GHI R15
E166  38                SKP
E167  FC   09           ADI $09
E169  9F        LETR:   GHI R15
E16A  FA   0F           ANI $0F
E16C  FF   00           SMI $00        ;CLEAR ERROR DF=1
```

```
E16E  C8                LSKP
E16F  FC  00   NOASC:   ADI $00        ;SET ERROR DF=0
E171  BF                PHI R15
E172  30  01            BR FINI +1
E174  E2       VTSTDE:  SEX R2         ; TEST RD > E
E175  8D                GLO R13
E176  52                STR R2
E177  8E                GLO R14
E178  F5                SD
E179  AF                PLO R15        ;KEEP RESULT
E17A  9D                GHI R13
E17B  52                STR R2
E17C  9E                GHI R14
E17D  75                SDB
E17E  30  00            BR FINI
E180  F8  13   VPNTAD:  LDI YOFF       ;POINT TO YOFFSET
E182  AC                PLO R12
E183  EC                SEX R12
E184  12                INC R2         ;POINT TO Y
E185  42                LDA R2         ;GET IT FROM STACK
E186  F4                ADD            ;ADD X OFFSET
E187  AA                PLO R10        ;MOVE IT TO R.A
E188  2C                DEC R12        ;POINT TO XOFFSET
E189  02                LDN R2
E18A  F4                ADD
E18B  52                STR R2
E18C  FA  07            ANI $07
E18E  AC                PLO R12
E18F  0C                LDN R12        ;GET BIT MASK
E190  AB                PLO R11
E191  02                LDN R2         ;GET X BACK
E192  FB  FF            XRI $FF        ;COMPLIMENT
E194  F6                SHR            ;& DIVIDE
E195  F6                SHR            ;BY 8
E196  F6                SHR
E197  52                STR R2
E198  F8  05            LDI $05        ;MULTIPLY BY 32
E19A  A8                PLO R8         ;THAT'S 1 LINE UP
E19B  F8  00            LDI $00        ;FOR EVERY Y POINT
E19D  BA                PHI R10
E19E  8A       MUL:     GLO R10        ;BEGIN MULTIPY
E19F  FE                SHL
E1A0  AA                PLO R10
E1A1  9A                GHI R10
E1A2  7E                SHLC           ;THIS GET'S ADDRESS OF Y
E1A3  BA                PHI R10
E1A4  28                DEC R8
E1A5  88                GLO R8
E1A6  3A  9E            BNZ MUL        ;LOOP TILL DONE
E1A8  F8  14            LDI VSTART     ;POINT TO VSTART
E1AA  AC                PLO R12
E1AB  E2                SEX R2
E1AC  8A                GLO R10
E1AD  F4                ADD            ;ADD X TO ADDRESS
```

```
E1AE   AA              PLO R10
E1AF   EC              SEX R12
E1B0   9A              GHI R10        ;ADD CARRY & VSTART
E1B1   74              ADC
E1B2   BA              PHI R10
E1B3   30  00          BR FINI
E1B5   F8  01  VNEG2:  LDI $01
E1B7   73              STXD
E1B8   C8              LSKP
E1B9   43      VNEG1:  LDA R3
E1BA   AC              PLO R12
E1BB   0C              LDN R12
E1BC   FD  00          SDI $00
E1BE   5C              STR R12
E1BF   30  01          BR FINI +1
E1C1   9F      VTOP:   GHI R15
E1C2   FF  C0          SMI $C0        ;TEST FOR OVER TOP
E1C4   38  00          BNF FINI       ;LEAVE D AS IS
E1C6   F8  BF          LDI $BF        ;LIMIT TO TOP
E1C8   30  01          BR FINI+1
     0D00              .= $0D00       ;BIT MAP
0D00   80  40  BITMAP: .WORD $8040
0D02   20  10          .WORD $2010
0D04   08  04          .WORD $0804
0D06   02  01          .WORD $0201
     0D10              .=.+8          ;DATA TABLE
0D10   00  00  DTAB:   .WORD $0000
0D12   00      XOFF:   .BYTE $00      ;XOFFSET
0D13   00      YOFF:   .BYTE $00      ;YOFFSET
0D14   E8      VSTART: .BYTE $E8      ;START ADDRESS OF VIDRAM
0D15   00      WRTFLG: .BYTE $00      ;WRITE FLAG
0D16   00              .BYTE $00
0D17   00  00  DELTX:  .WORD $0000    ;DELTA X & SIGN
0D19   00  00  DELTY:  .WORD $0000    ;DELTA Y & SIGN
0D1B   00  00  ERTRM:  .WORD $0000    ;ERROR TERM
0D1D   00  00  LAST:   .WORD $0000    ;LAST ADDRESS TO DO
0D1F   00      MASK:   .BYTE $00      ;LAST BIT TO DO
     0000              .END
```

The WD2412 Time of Day Clock

- by D. Schuler, 3032 Avon Rd., Bethlehem, PA. 18017

After having read many articles on real-time clock chips in Byte, Electronics Design News (EDN), Electronics, and of course Ipso Facto, I began to look for a device to be incorporated into my system. My major objective was to look for a device that could be accessed by a read or write instruction in only one bus cycle, with NO wait states required. The only device that I found that fit this description was the Western Digital WD2412 Time of Day Clock. This device has an access time of only 250ns, an alarm, and built-in control for a multiplexed LED time/date/user display. It can provide the time on one of three formats: 12 hour, 24hour, or binary form, and can provide interrupts from once every 0.1 second to once every day.

Hardware

The required circuit to connect the 2412 to an 1802uP system is shown in Figure 1. This circuit has been connected so as to reset itself only when power is first applied to the system. If you desire to reset the counter whenever the CPU is reset, delete the 3.3K resistor and 47uF capacitor, and instead connect the master reset ($\overline{MR}$ - pin 4) to the system reset line. An interrupt request line is also available at pin 17, which can be used to signal the 1802 that servicing of the 2412 is required. This line makes a low-to-high transition when the interrupt register needs servicing, but it must be latched until the 1802 has received the interrupt.

Figure 2 shows an optional display circuit that can be used to display the time (in 12 or 24 hour mode), date, or a user register. The display is controlled entirely by the 2412, which provides conplete control of the multiplexed LED display without any processor intervention.

Software

The WD2412 will accept and execute 24 commands from the 1802 microprocessor. When the device is ready to accept a command, the status register will contain EEH. Once a command has been received by the 2412 and a data transfer takes place to the 1802, the 1802 must acknowledge the transfer by sending FFH to the 2412. If a commans sequence is not completed within one second, the 2412 will terminate the command and any data transferred will be disregarded.

The commands of the 2412 can be divided into three basic types (see Table 1). Type I commands are used to set internal flags and operating modes, and are complete after the transfer has been received by the 2412 TDC. In Tiny Basic, the display would be set to show the date by the following sequence:

```
        .
        .
        .
100 LET A = (address of 2412)
110 IF PEEK(A) <> 238 GOTO 110       ..wait until ready for
                                     ..command
120 POKE A,227                       ..E3H
        .
        .
        .
```

Type II commands are used to set internal registers of the 2412. All of these are followed by three bytes of data, and the commands will not be interrupted by an internal interrupt from the 2412. A routine to set the date to January 10, 1985 (01/10/85) would be as follows:

```
          .
          .
          .
100 LET A = (address of 2412)
110 IF PEEK (A) <> 238 GOTO 110      ..wait until ready
120 POKE A,235                       ..EBH
130 IF PEEK (A) <> 255 GOTO 130      ..wait for acknowledge
140 POKE A,133                       ..byte 1: year - 85H
150 IF PEEK (A) <> 255 GOTO 150      ..wait for acknowledge
160 POKE A,16                        ..byte 2: day - 10H
170 IF PEEK (A) <> 255 GOTO 170      ..wait for acknowledge
180 POKE A,1                         ..byte 3: month - 01H
          .
          .
          .
```

The Type III commands are used to read the internal registers of the 2412 TDC. The 2412 follows each of these commands with three bytes of data, and it will not generate an interrupt during one of these commands. A routine to read the time in a 12 hour BCD format would be as follows:

```
          .
          .
          .
100 LET A = (address of 2412)
110 IF PEEK (A) <> 238 GOTO 110      ..wait until ready
120 POKE A,240                       ..F3H
130 PR"SECONDS = "; PEEK (A)         ..byte 1: seconds
140 POKE A,255                       ..acknowledge receipt
150 PR "MINUTES = "; PEEK (A)        ..byte 2: minutes
160 POKE A,255                       ..acknowledge receipt
170PR "HOURS = "; PEEK (A)           ..byte 3: hours
180 POKE A,255                       ..acknowledge receipt
          .
          .
          .
```

Internally the 2412 TDC contains eight 3-byte user addressible registers: 12 hour BCD time of day (BCD12), 24 hour BCD time of day (BCD24), binary time of day (BIN), date (DATE), day (DAY), user (USR), and two interrupt (INT) registers. The format of these is as follows:

```
                   byte 3                    byte 2                    byte 1
BCD24 - [  H10   ][   H1   ]  [  M10   ][   M1   ]  [  S10   ][   S1   ]
         7 6 5 4  3 2 1 Ø     7 6 5 4  3 2 1 Ø     7 6 5 4  3 2 1 Ø

BCD12 - [PM][ H10 ][   H1   ]  [  M10   ][   M1   ]  [  S10   ][   S1   ]

BIN   - [ØØ bits][14 - 19]   [Ø  bits][ 7 - 13]   [Ø  bits][ Ø - 6 ]

DATE  - [  M10   ][   M1   ]  [  D10   ][   D1   ]  [  Y10   ][   Y1   ]

USER  - [   D5   ][   D4   ]  [   D3   ][   D2   ]  [   D1   ][   DØ   ]

DAY   - [Ø Ø Ø Ø][Ø Ø Ø Ø]   [Ø Ø Ø Ø][Ø Ø Ø Ø]   [Ø Ø Ø Ø][  DAY   ]

INT   - relative: same as binary register
        absolute: same as BCD24 register
```

A sample routine to initialize the Wd2412 is shown in Listing 1 and the RUN of this is shown in Listing 2. These show how the 2412 can be controlled from a program in BASIC and a useable form of information can be obtained.

A complete data sheet for the WD2412 TDC can be obtained from Western Digital Corporation, 2445 McCabe Way, Irvine, CA 92714.

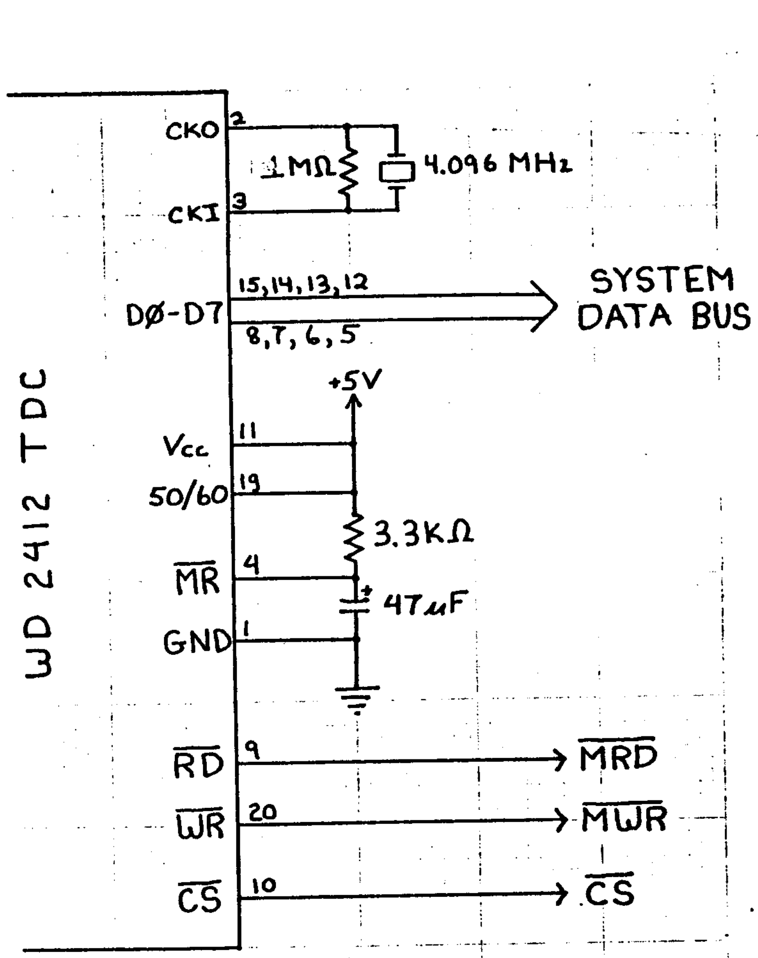


FIGURE 1

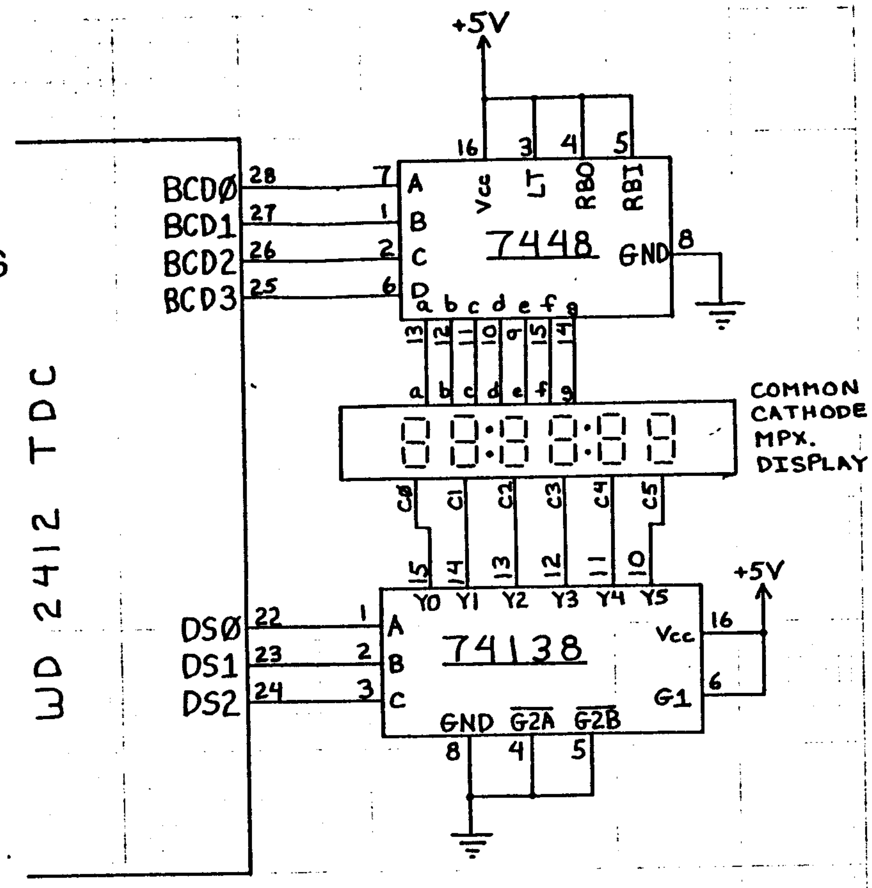


FIGURE 2

SUMMARY OF COMMANDS

Type I Commands

E∅H - Display off
E1H - Display 12 hour time
E2H - Display 24 hour time
E3H - Display date
E4H - Display user
E5H - Select crystal timebase
E6H - Select external 50Hz timebase
E7H - Select external 60Hz timebase
E8H - Disable interrupts

Type II Commands

E9H - Set 12 hour BCD time
EAH - Set 24 hour BCD time
EBH - Set date
ECH - Set day of week
EDH - Set user
EFH - Set absolute 12 hour time interrupt
F∅H - Set absolute 24 hour time interrupt
F1H - Set absolute date interrupt
F2H - Set relative interrupt

Type III Commands

F3H - Read 12 hour BCD time
F4H - Read 24 hour BCD time
F5H - Read binary time
F6H - Read date/power fail flag
F7H - Read day of week
F8H - Read relative interrupt

TABLE 1

```
:LIST

10 REM SAMPLE PROGRAM FOR:
20 REM WD2412 TIME OF DAY CLOCK
30 REM
40 REM WRITTEN BY DAVID W. SCHULER
50 REM               3032 AVON ROAD
60 REM               BETHLEHEM, PA   18017
70 REM
80 REM ADDRESS OF 2412 IN MY SYSTEM
90 LETA=49152
100 REM TURN ON DISPLAY
110 REM 'DISPLAY 12 HOUR TIME - E1H'
120 REM CHECK STATUS
130 IFPEEK(A)<>238GOTO130
140 REM WRITE COMMAND
150 POKEA,225
200 REM SET TIME OF DAY (24 HOUR MODE)
210 REM 'SET 24 HOUR BCD TIME - EAH'
220 PR"ENTER TIME: HH,MM,SS",
230 INPUTX,Y,Z
240 REM CONVERT DECIMAL TO BCD
250 LETX=((X/10)*16)+(X-((X/10)*10))
260 LETY=((Y/10)*16)+(Y-((Y/10)*10))
270 LETZ=((Z/10)*16)+(Z-((Z/10)*10))
280 REM WRITE TIME TO 2412TDC
290 IFPEEK(A)<>238GOTO290
300 POKEA,234
310 IFPEEK(A)<>255GOTO310
320 POKEA,Z
330 IFPEEK(A)<>255GOTO330
340 POKEA,Y
350 IFPEEK(A)<>255GOTO350
360 POKEA,X
370 REM
400 REM 'READ BINARY TIME - F5H'
410 REM READ TIME
420 IFPEEK(A)<>238GOTO420
430 POKEA,245
440 LETZ=PEEK(A)
450 POKEA,255
460 LETY=PEEK(A)
470 POKEA,255
480 LETX=PEEK(A)
490 POKEA,255
500 REM FORMAT DATA
510 LETZ=Z+(128*(Y-((Y/2)*2)))
520 LETY=(Y/2)+(64*(X-((X/4)*4)))
530 LETX=X/4
540 REM WRITE BINARY TIME
550 PR"BINARY TIME: ",X,Y,Z
560 PR"END OF SAMPLE PROGRAM"
570 END
:
```

LISTING 1

```
:RUN

ENTER TIME: HH,MM,SS   ? 15,36,12

BINARY TIME:   8      146      58
END OF SAMPLE  PROGRAM

:
```

LISTING 2

Centronics Parallel Port for the Super ELF

- by G. Jones, 7717 N. 46th Drive, Glendale, Arizona 85301

Recent issues of IPSO FACTO have had a couple of articles on adding a "Centronics" type parallel output port to your 1802 micro. Each of these articles has its merits. However, each author started from "ground zero" by adding all the components required for the port, including the PIA (Parallel Interface Adapter) chip.

This approach is fine for those who own an Elf II or other fine 1802, but it's not necessary to go the whole distance if you own a Quest Super Elf. This great little machine has a lot of features which have had to be added to the other 1802 micros. In this present instance, the Quest 4K Super Expansion Board already has designed into it a pair of parallel I/O ports. U50 is wired directly to a 16 pin dip input header (Input Port 5), and U51 is wired to the output header (Output Port 3).

My first I/O set-up consisted of the 1861 Pixie chip and a video monitor for output, and a GRI parallel ASCII keyboard on Port 3 for input. Eventually I got tired of Pixie-Graphics text, and got myself a Netronics terminal board to hook up to the serial I/O ports. I never did get around to implementing the parallel output port until this past Christmas, when ol' Santa brought me an Okidata 82A printer.

I immediately stuffed an 8212 PIA chip into U51 on the expansion board, and after a little work with a soldering iron and some ribbon cable, I plugged my new toy into the output port and loaded Super Basic, because it already has a parallel printer driver in it. Next, I wrote a short demo loop in Basic and typed RUN. The printer sat there mutely, and the Elf hung up. Oh well, back to the drawing board.

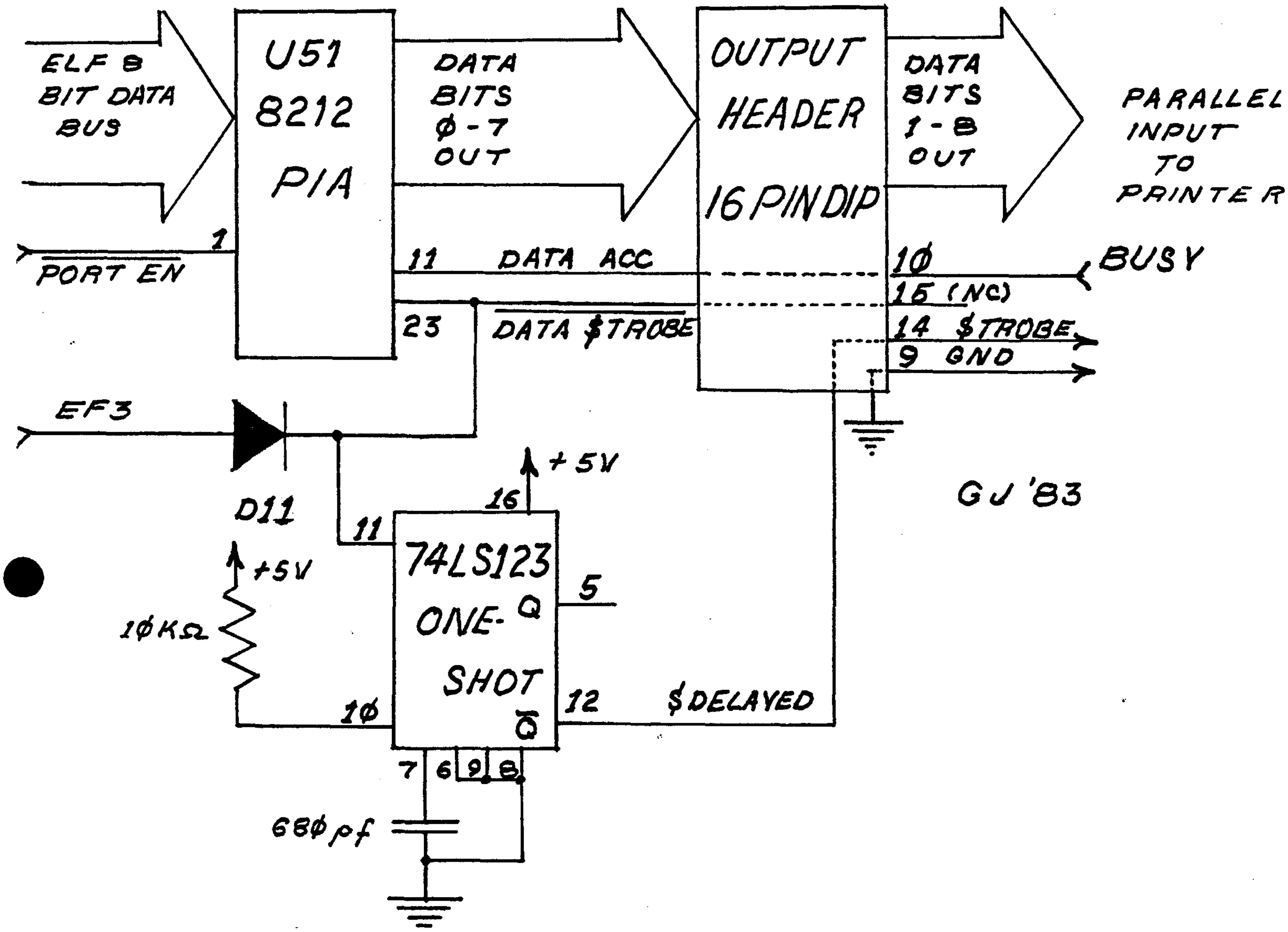
Each of the articles in IF shows some method of delaying the data strobe, with a one-shot and an RC network. Examination of the Expansion board schematic showed an output at pin 23 of the 8212 labled "STB" and "Data Avail". I borrowed a Centronics 730 Manual and an o'scope and began digging. The manual showed a square little 1 usec negative going strobe neatly centered in the middle of a 3 usec data pulse.

The borrowed 'scope showed a data pulse of about 8 usec at each data pin on the Elf output header, and a 3 usec strobe at output header pin 10 which had a .5 usec head-start on the data. Using a 74LS123 one-shot and some junk box parts, I soon had a neatly centered 3 usec strobe following the start of the data pulses, and an Okidata printer happily singing away as it PLIST-ed a program.

The last item on the agenda was to write a parallel printer driver to use with my Editor/Assembler. It's short, sweet, and completely relocatable.

While the port circuits already seen in IF may work just fine in other micros, there's no need for us Super Elf owners to ignore that perfectly good output port already designed into our Expansion boards.

SUPER ELF
"CENTRONICS" PARALLEL INTERFACE



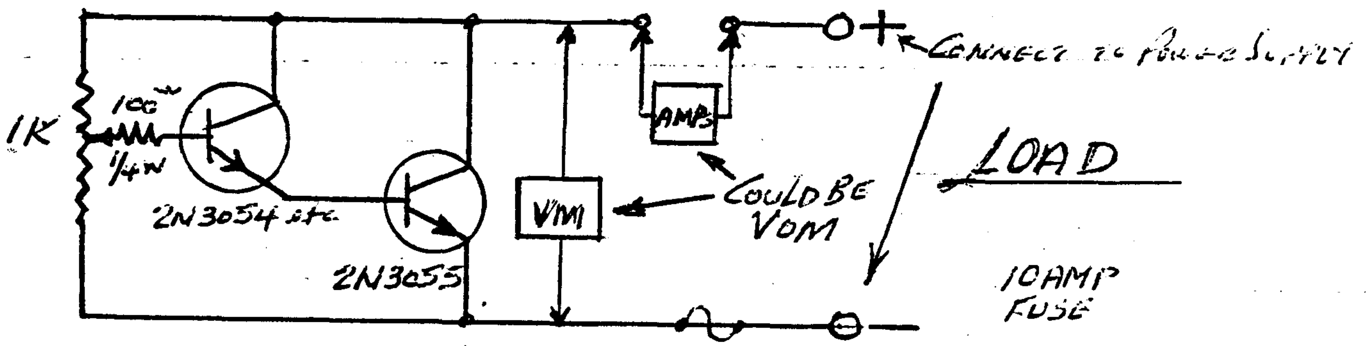

```
0000                 ORG #0000
0000 9F      PRINT :GHI RF;  GET THE OUTPUT CHARACTER
0001 52              STR R2;  SAVE IT ON THE STACK
0002 E2              SEX R2; AND MAKE SURE R2 IS THE X REGISTER
0003 63              OUT3;  SO THE OUTPUT PORT 3 INSTRUCTION CAN FIND IT
0004 22              DEC R2;  RESTORE THE STACK
0005 3E05    PRT.1: BN3 PRT.1;  THEN WAIT TIL THE PRINTER IS DONE
0007 D5              RETN
0008         ;
0008         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0008         ;
0008         ;          THIS ROUTINE ASSUMES PORT 3 AS THE OUTPUT
0008         ;          PORT AND EF3 AS THE INPUT FLAG.  ENTER THE
0008         ;          SUBROUTINE AT "PRINT" USING 1802 SCRT PROTOCOL
0008         ;          AND EXIT VIA THE SEP R5 AT THE END.  IT CAN BE
0008         ;          RELOCATED ANYWHERE IN MEMORY.
0008         ;
0008         ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Electronic Power Supply Load Tester
- by Fred Feaver

How many of you have wished you knew exactly how much current your power supply could supply without collapsing, or just how good your voltage regulation was?

Here is a simple, inexpensive load tester that is very easy to build. The total cost should be much less than $5.00. With a 10 ampere capacity at up to 40 volts, this will handle practically all hobby computer supplies.

Note: The quoted price is for the electronic components.



The 2N3055 has a 60 volt rating at 15 amps. H_{FE} = 20 - 70 Design for 10 amps. Take worst case H_{FE} = 20

$$\text{Base Current } I_B = \frac{Ic}{H_{FE}} = \frac{10}{20} = 0.50 \text{ amps}$$

It is possible to connect the base directly to a potentiometer, but it is better to use a Darlington configuration to reduce the current through the pot.

Say the driver transistor has H_{FE} of 20 with 0.5 amp output:

$$\text{then} \quad I_B = \frac{0.5}{20} = 25 \text{ ma.}$$

Almost any small or medium NPN transistor can be used. I used an obsolete 2N696 from my junk box, it had an output current rating of 500 ma and H_{FE} of 20.

NOTES ON THE DON STEVEN'S NOTABLE ASSEMBLER

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

NOTABLE ASSEMBLER: This is an excellent tool for small machines. I keep my source code in a notebook, leaving blank lines for adding code. Since no listing is printed, this is necessary. My Selectric printer takes about 5 minutes per page, so I do not want to print long listings, and find the NOTABLE ASSEMBLER a fine programming tool. I wrote to Don for more information, and he sent the following, which may be of value to others using the assembler.

SOURCE	OBJECT	DESCRIPTION
042F	011B	Call routine to put immediate data on R7 stack
043B	0124	Call Search routine, look for 68 04
0456	013A	Call routine to get address of the 6804xxyy
0462	0144	Call routine to look for the xxyy following 6804 xxyy in source, and get object address
049A	0169	All done with 6804 stuff
04B2	017D	Main loop. Check if at end of source, if yes, go to 05B5 (source), 01FE (object), else look at next byte
0532	01C0	Call routine to put object byte in memory
0537	01C3	Go to main loop
05B5	01FE	Tidy up and return (at 0215)
0638	024A	Routine to search for a name, then get its object address
06C3	02A2	Routine to get object address, knowing source address
07D3	0327	Routine to put byte in memory (object)
07E8	0338	Routine to make R7 stack from immediate data
082F	035D	Routine for saving regs, restoring regs, and processing errors (not SCRT, called by SEP E)
0874	03DE	Routine which finds a pattern in source corresponding to R7 pattern stack

If you do not want the 68 3j questionable references put on the R8 stack, change byte at 01EE (source = 058E) from 3A to 30.

After using the assembler for some time, I have found a couple of quirks. Sometimes a 68 4j will result in a long branch when it could have been a short branch. This is of little consequence unless you are making a source program from an existing object program. Of more concern is that the 68 4j works by changing the byte prior to the 68 4j from xy to 3y for a short branch, or Cy for a long branch. Since the long and short branch sets are not symmetrical, this can result in a problem. For example, 34 is a B1, while C4 is a NOP. For this reason, the 34, 3C, 35, 3D, 36, 3E, 37, 3F, C4, CC, C5, Cd, C6, CE, C7, and CF op-codes should never be followed by 68 4j. The main usage problem I have is in failing to put 00 after a 68 which is to be placed in the object code. This can create all sorts of problems if followed by 01, 02, 03, 1j, etc.

A Simple Controller

- by Harley Shanko, 15025 Vanowen, #209, Van Nuys, Ca. 91405 USA

The following information may be of interest to ACE members.

A Simple Controller

For those who may require a relatively simple, compact, low-cost controller, this may seem heretical, but the National INS8073 cpu with on-chip 2.5 K integer BASIC is good choice; the Zilog Z8 is possibly a bit more powerful (and costly) hardware-wise, although National's BASIC is better. It includes (110, 300, 1200 and 4800 baud) software UART, interrupt sensing, DO...UNTIL, FOR...NEXT, DELAY (1 - 1040 ms), MODulo function, RND, TOP (top-of-program 'pointer' for RAM usage - buffers, stacks, etc; string capabilities, power-on/reset RUNs if BASIC in EPROM is at #8000, just to name a few.

I have built two configurations; they have been very simple to make operational. A minimal configuration is shown, Fig ___; adding a terminal (I use an RCA VP3301 with a TV) and a power supply (< 150 ma @ +5v), programs can be generated and run. Of course, controller applications would typically require additional I/O, possibly an EPROM (turn-key program). A EPROM programmer requires very little more than a +25 v power supply and a 50 ms one-shot circuit.

The INS807x can be used in multiprocessor applications; so the busses are 3-state except when external memory accessed, and 3 control lines are used to provide arbitration. Several companies, Transwave, Octagon, Essex - of England, use the -8073 and make 4.5 x 6.5 inch sized pc boards. For more information these references are listed:

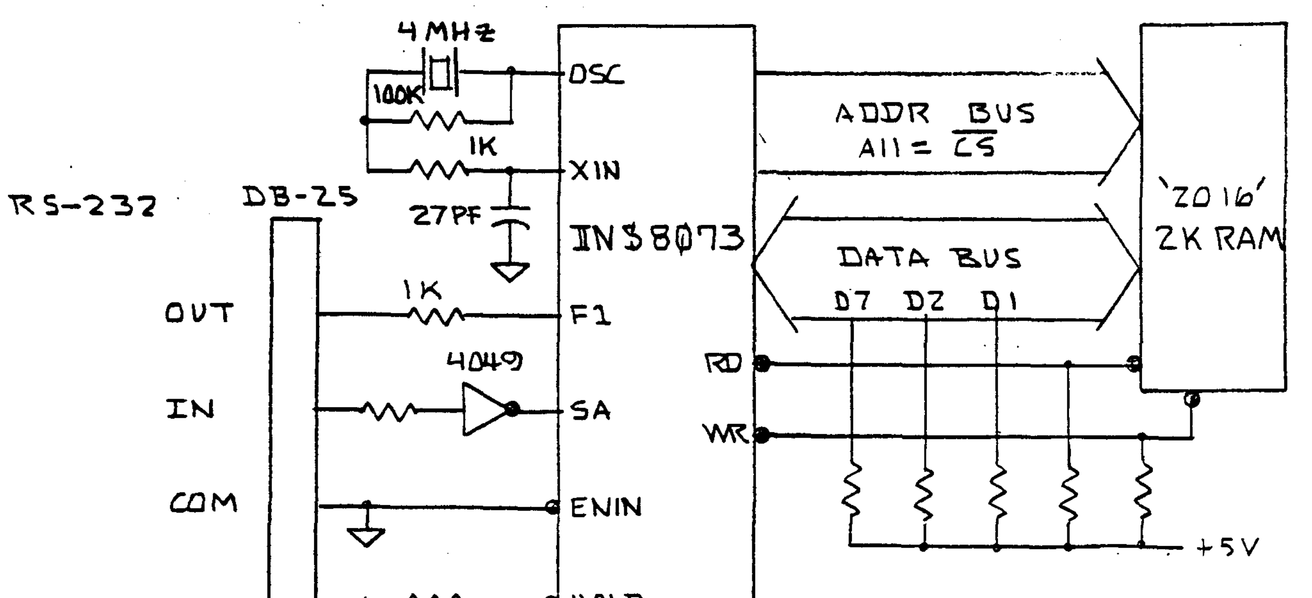
1. National Semiconductor Corp:
   Data sheet, Oct 80, 20 pgs, "INS8070-series Microprocessor Family".
   Data sheet, Feb 81, 4 pgs, "INS8073 NSC Tiny BASIC Microinterpreter".
   User manual, Nov 80, 190 pgs, "NSC Tiny BASIC", publication number 420306319-001A.
   User manual, 20 pgs, "Using NSC Tiny BASIC.
2. EDN, Aug 80, pg80+, "1-chip uC's, high-level languages combine for fast prototyping".
3. Electronic Design, Nov 22, 80, pg235+, "On-chip Tiny BASIC dumps development systems".
4. Electronic Design, Oct 15, 81, pg 237, "One-Chip uC, two RAMS, and an op-amp makes versitile computer system".
5. Microcomputing, Dec 81, pg32+, "Everyman's Computer System".
6. BYTE, Apr 82, pg472+, "An Introduction to NSC Tiny BASIC".
7. EDN, Feb 3, 83, pg 77+, "BASIC-equipped single-board computers function as low-cost controllers".

Speech Synthesizer

For those who need or can use a speech synthesizer, the unit referred to by Steve Ciarcia in BYTE, Sep and Oct 83 issues, provides much versitility; I feel it exceeds Votrax's 'Type & Talk' unit in capability, even though both utilize the SC-01 synthesizer IC. Intex Micro Systems Corp sells the assembled version, $295, and Micromint, Inc the kit version, $215.

The unit has both 'Centronics' parallel and RS-232 serial interface. I can be programmed easily to spell each letter, or pronounce text; to pronounce some, most or all punctuation; has phoneme capabilities which allows forming almost any conceivable speech sounds/words; can generate musical tones; has a 750 character text buffer, a 6 K text-to-phoneme algorithym, 64 inflection levels, and a built in speaker; about 20 control-code sequences provide the programability.

I have used this unit with a TRS80-II, my home brew 1802 and 8073 systems. It is very simple to interface; can tie it directly to a terminal and type in letters, words, sentences, and commands. The quality of speech is more than adequate for many applications; however, be advised that presently NO low-priced speech synthesizer can pronounce every word precisely the way you may think it should be. The high-quality ones need many K-bits/sec of data, thus require much RAM or EPROM; whereas, the lower-quality schemes attempt to use less than 1-2 K bits/sec (the TI 'Speak-N-Spell' for instance).



BR	D2	D1
110	1	1
300	1	0
1200	0	1
4800	0	0

Fig 2. Minimal INS8073 system

CHIP 8 Game Mods for the 6847
- by M. Franklin, Milton, Ontario

Tony Hill's article on modifying the RCA CHIP 8 and VIP Operating system to drive the ACE 6843 VDU was a real boom to my kids and I. The CHIP 8 games provided my family with hours of fun and challenge. My one regret with "trading" up to the ACE CPU was the loss of my 1861 "big" graphics from Netronics Tiny Basic and the CHIP 8 games. Tony's article redresses that loss. Unmodified CHIP 8 games run quite well on the new CHIP 8 ACE-VDU driver and 6847, but are often difficult to watch or play because of the new "large" screen of the 6847. The game area will appear in the upper left quarter of the screen, but the "players" may roam all over the screen, often outside of "control space". Also, the 6847 2 x 1 bit format shape makes some of the graphics difficult to read. In addition to these problems, people now using a keyboard may find the hex pad control numbers configuration difficult to use. Finally, one additional mod I have made to my game is that "control key" reset at end of game, where by the program waits for a key press of any character then reinitializes the game instead of spinning in a closed loop.

The following examples from games in the VIP 311 manual illustrate the changes necessary to implement the above.

Size of Playing Field

The user has 3 choices - full screen, centre screen or half screen, depending upon game play field configuration. Some games - such as TIC TAC TOE or ACEY DUECY have fixed play areas and are best centered on the screen, others such as ARMOURED VEHICLE CLASH can use the whole screen. The programs use wrap checks on the X and Y variables to see if they exceed the playing field area - X = 3F max, Y = 1F max (64 x 32). The VDU has a 7F (X) by 3F (Y) playing field available (128 x 64). To find the appropriate variables - look for a DXYN Instructions, and find the SKIP VX = (NE) 3F (4X1F) or (4Y1F) instructions. Change appropriately: ie. #12 Armoured Venicle Clash Addresses:

0384-4338 change to 4378
0388-4418 change to 4438
0398-433F change to 437F
039C-441F change to 443F

To centre the TIC TAC TOE display (#7), change:

addresses	02B4	from	1401	to	3411		02BE	from	2409	to	4419
	02B6	from	1C01	to	3C11		02C0	from	1411	to	3421
	02B8	from	2401	to	4411		02C2	from	1C11	to	3C21
	02BA	from	1409	to	3419		02C4	from	2411	to	4421
	02BC	from	1C09	to	3C19						

to change play box area, and address:

0332	from	6014	to	6034
0334	from	6100	to	6110
034E	from	601C	to	603C
0350	from	611A	to	6130

In ACEY DUECY (#20), change the following to centre the display:

addresses	020A	from	6113	to	6130
	0210	from	6127	to	6150
	0234	from	611D	to	6140
	028C	from	6318	to	6338
	0294	from	7306	to	7308
	029A	from	7306	to	7308
	02B4	from	6200	to	6208
	02B8	from	D127	to	D129
	02BC	from	7101	to	7102
	0310	from	60FC	to	60FF
	0312	from	FCFC	to	FFFF
	0314	from	FCFC	to	FFFF
	0316	from	FCFC	to	FFFF
	0318	from	--	to	FFFF

Keyboard Command Code

To change the command keycode, first decide on a more appropriate configuration - I use 1 left 2 up 3 right 4 down 5 go or fire 6 cancel instead of 6 left 2 up 4 right 8 down F fire.

#12 - change	0204	-	6A08	to	6A04
	0206	-	6906	to	6903
	0208	-	6804	to	6801
	0296	-	6001	to	6005

TIC TAC TOE and ACEY DUECY do not require changes since they use all keys for values 0 - F hex.

Game Reset

To reset the game, locate the exit closed loop - usually a GO to self ie. 1282, and change to GO - spare address and add a routine that waits for a key press then clear screen and jumps to 0200 (h) - FX0A
00ED
1200

In #12 address -	0364	from	1364	to	1480
	0480	FEoA			
	0482	00E0			
	0484	1200			
In #7 - address -	0244	from	1356	to	1354
	0354	F30A			
	0356	00E0			
	0358	1200			
In #12 - address -	1282	from	1282	to	131A
	031A	from	F30A		
	031C	from	00F0		
	031E	from	1200		

HAVE FUN!

Submarine - A CHIP 8 Game

Press 1 to drop depth charge on submarine,

M(020A) = # of shots, M(0241) = # of subs sunk to win

```
0200   68 00 6D 00 6C 00 61 0C 22 C2 6E 19 22 E6 62 00
0210   A3 AF D2 11 72 08 32 80 12 12 41 1F 12 24 61 1F
0220   62 00 12 12 61 30 62 08 64 0A 65 20 66 15 6B 01
0230   A3 AA D1 24 48 01 23 06 48 00 22 F4 C7 04 85 74
0240   4D 05 14 50 4E 00 13 16 A3 AE D5 62 34 0A 12 5A
0250   EB 9E 12 62 44 0A 64 0D 83 10 D3 41 6C 01 4F 01
0260   12 9C 6A 10 8A B5 3A 00 12 64 4C 01 12 7A A3 AE
0270   D5 62 A3 AA D1 24 81 B5 12 30 D3 41 74 01 D3 41
0280   4F 01 12 9C D3 41 74 01 FB 18 34 1F 12 6E FB 18
0290   22 E6 8E B5 22 E6 6C 00 64 0A 12 6E D3 41 D5 62
02A0   14 00 64 08 44 00 14 36 84 B5 62 06 13 72 22 C2
02B0   7D 01 22 C2 22 E6 8E B5 22 E6 FA 18 64 0A 6C 00
02C0   12 72 A3 F0 FF 55 64 6C 6C 01 A3 E0 FD 33 F2 65
02D0   F0 29 D4 C5 74 07 F1 29 D4 C5 74 07 F2 29 D4 C5
02E0   A3 F0 FF 65 00 EE A3 F0 FF 55 64 03 6C 01 A3 E3
02F0   FE 33 12 CE 89 60 C7 01 89 74 49 1E 13 02 86 74

0300   00 EE 68 01 00 EE C7 01 89 75 49 0C 13 12 86 75
0310   00 EE 68 00 00 EE A3 AA D1 24 61 28 D1 24 61 0A
0320   62 10 A3 AE D2 12 72 08 D2 11 4F 01 13 34 D2 11
0330   72 01 13 28 D2 11 62 08 61 28 A3 AA D1 24 D1 24
0340   42 1B 13 4A D1 24 72 01 13 3E 61 24 62 20 A3 B0
0350   D2 15 72 08 A3 B5 D2 15 72 08 A3 BA D2 15 72 08
0360   A3 BF D2 15 72 08 A3 C4 D2 15 72 08 A3 C9 D2 15
0370   14 36 6E 00 A3 D0 D1 21 7E 01 3E 10 13 78 D1 21
0380   82 B5 32 00 13 72 FB 18 62 00 A3 CE D1 23 23 E6
0390   A3 D1 D1 24 23 E6 A3 D5 D1 25 23 E6 A3 DA D1 25
03A0   23 E6 7E 01 3E 12 13 86 12 A4 10 18 FE 7E 30 FF
03B0   44 28 10 10 10 78 48 48 48 78 90 90 90 90 F0 81
03C0   81 81 81 F1 E3 22 23 20 E3 C7 04 C7 44 C7 00 00
03D0   10 00 28 44 28 10 00 82 00 10 54 00 00 00 54 00
03E0   00 00 05 00 01 02 63 0A 43 00 00 EE 83 B5 13 E8
03F0   00 07 08 41 14 84 0C 04 01 0D 00 01 00 05 0C 01

0400   63 30 64 14 A4 3C D3 45 73 09 A4 41 D3 45 73 08
0410   A4 46 D3 45 6C 20 FC 15 FC 07 4C 00 14 20 14 18
0420   63 30 64 14 A4 3C D3 45 73 09 A4 41 D3 45 73 08
0430   A4 46 D3 45 12 AE FA 0A 00 E0 12 00 F1 51 71 51
0440   F1 C7 45 45 45 C7 11 1B 15 11 11 00 00 00 00 00
0450   D1 24 61 1C D1 24 63 24 64 30 A3 B0 D3 45 73 08
0460   A3 B5 D3 45 73 08 A3 BA D3 45 73 08 A4 80 D3 45
0470   73 08 A4 85 D3 45 73 08 A4 8A D3 45 12 A2 00 00
0480   88 88 A8 D8 88 E2 43 42 42 E2 22 22 A2 60 22 00
```

MICROPROCESSOR CONFERENCE '83

THE ASSOCIATION OF COMPUTER
CHIP EXPERIMENTERS INC.
PRESENTS

THE TECHNICAL AND PRACTICAL
IMPLEMENTATION OF
MICROPROCESSORS

FEATURING THE RCA
CD1802/4/5/6 CMOS MICROS

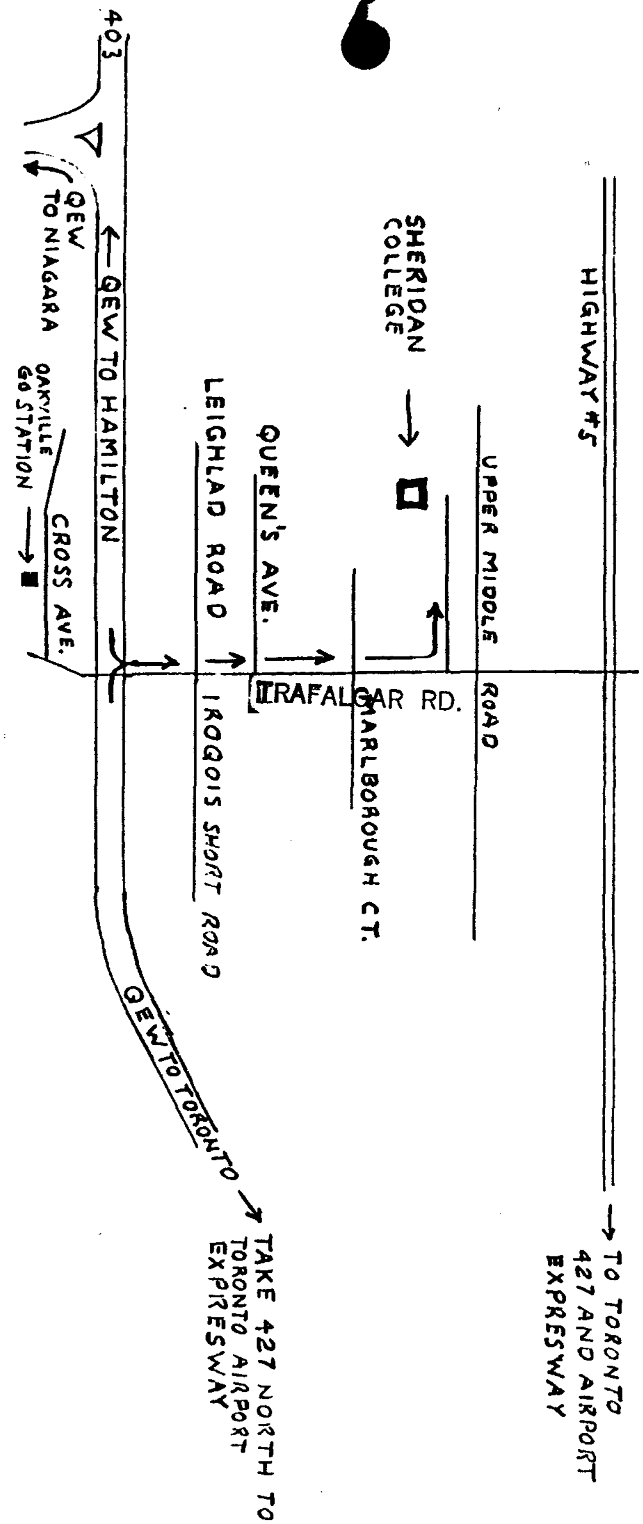
AT

SHERIDEN COLLEGE
TRAFALGAR CAMPUS
IN SHERIDAN HALL
SATURDAY 20th of AUGUST 1983
COMMENSING at 9:00AM

PROGRAM
*TELIDON VIDEOTEX
*AMATEUR PACKET RADIO
*CONTRL SYSTEMS
*PROGRAMING
*INTERFACING
*DISPLAY OF MICROPROSSER
*EQUIPMENT

****
* PRIZES PRIZES PRIZES *

* RCA 3800 Series Teletex *
* Terminal with Built in *
* Memory,Vidio Modulater *
* and Modem Comes with *
* one free hour of time *
* on COMPUSERVE *
****



QUESTIONAIR

Primary job function...

Indusrery..

Products...

Type of Computer system......................................

CPU..

Memory...

Peripheral equipment...

Would you like to give a short talk
on what you are doing with micros?...........................

PROGRAM

8:30 AM REGISTRATION

9:00 TELEDON TELETEX VIDIOTEX by Jim Greer Sheridan computer systems

10:00 COFFEE

10:15 APPLICATION OF MICROS IN CONTROL SYSTEMS.

11:00 MULTI-CHANNEL A to D CONVERSION by Don MacKenzie.

12:00 pm LUNCH

1:00 PROGRAMMING WITH FORTH.

2:00 INTRODUCTION TO PACKETT DATA TRANS-MISSION USING AMATEUR RADIO by VE3 NEC John Langtree

3:00 COFFEE .

3:15 APPLICATIONS OF MICROS AND SUPPORT CHIPS. Pierre Andeweg RCA, Field Applications Engineer.

4:15 MICRO FORAM SHORT TALKS

FOR MORE INFORMATION
CALL FRED PLUTHERO
(416) 389 4070

INTRODUCTION

The association of Computer Chip Experimenters Inc. was organized in 1977 by a group of people who were attending a course on Micro Computer Technology and Applications. The course was run by Eugene Tekatch at the request of The Institute of Electrical and Electronic Engineers. He used the RCA 1802 m.p.u. and by adapting products from his company's industrial controller line designed the TEC 1800 system. Some 600 people attended the course in the first few months and a large group of them showed an interest in forming a users group. As a result, Eugene and the group of enthusiastic helpers produced the first news letter consisting of 12 pages and sent it to all the course participants. The popularity of the newsletter prompted a second issue and a call to firm a club .

ACE has grown to more than 575 members, has become a club of international stature, and publishes the most significant source of 1802 micro processing user-information in the world. The club newsletter, Ipso Facto, is published six times a year and averages 50 pages per issue.

NO.

Do not use

REGISTRATION FOR MICROPROCESSR CONFERENCE '83'

Register by mail or at door. Tickets for pre-registrations will be at door.

NAME........................
ADDRES......................
CITY.........PROV/STATE...............POSTAL OR ZIP CODE.....

Registration Fee 25.00 No. Attending ......
Includes lunch Total Enclosed.....
and coffie
ACE Member 18.00

Make certified cheque or mony order payable to
THE ASSOCIATION OF COMPUTER CHIP EXPERIMENTERS INC.
C/O MIKE FRANKLIN 690 LAURIER AVENUE
MILTON, ONTARIO L7T-4R5