Ipso Facto

ISSUE 41 July 84

INDEX PAGE

A Publication of the Assosiation of Computer-chip Experimenters

Executive Corner	2
Editor's Corner	3
Member's Corner	3
Review - PASCAL with your Basic Micro	5
ACE Hardware - Pleasures, Problems, and Suggestions	8
ELF II Input Key Debounce	10
Forth Recursion - an Update	12
Calendar Maker	13
The ELF Tells Time	19
Chip 8 Disassembler	37
Club Communique	41

IPSO FACTO is published by the ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS (ACE), a non-profit educational organization. Information in IPSO FACTO is believed to be accurate and reliable. However, no responsibility is assumed by IPSO FACTO or the ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS for its use; nor for any infringements of patents or rights of third parties which may result from its use.

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

Well, hat in hand, I beg your forgiveness for publishing an article with so many errors or omissions in it, as Larry Owen's CHIP 8AE article contains at least 2 typos, no reference to port and flag corrections required for different systems, and 3 omissions. The corrections are as follows:

```
                10EA - 38 instead of 3B
                14DC - 3A            30
add at 11FA - D4 66 02 80 23 23
       1380 - E3 61 00 F8 CB A9 06 59 D4 24
       14D7 - 92 73 9F FB DF 3A D7 30 24     - VDU Version only
       14EA - 30 D7 E2 E2                    - VDU Version only
       14F5 - 00 00 00 00 00 00 00           - VDU Version only
delete 159D - C4                             - VDU Version only
move   1585 to 159C to 1586 - 159D           - VDU Version only
add at 1585 - C4
```

1861 Version Port References are located as follows:

```
       1381 - 61 turn off display
       1417 - 61    "  "    "
       14D3 - 61    " on    "
```

Flag 4 is used in this version:

```
15B5 - 3C
15BE - 3C
15C5 - 34
```

1861 version uses Inport 4 for the keyboard at 148E, while the VDU version uses Inport 7 at 148D.

All regular CHIP 8 games will run on this version, loaded at 0200h.

Hardware News

The version 11 Disk Controller Board is in production, and the members who have waited patiently so long for the new board should receive them by the end of August.

Work has statred on the 80 x 25 Video board, Tony Hill has a working proto type, which now has to be translated into artwork and a finished board.

The club is contemplating a D to A ; A to D board for contoller projects. I would appreciate hearing from you if you are interested in such a board, or if you have circuits which work for either function. Jim Kreter, from Worthington Ohio, is leading the design on this project.

Member's Corner

Al Magnani asks whether Eproms and Rams can be mixed on a single board, such as ACE's Eprom Board. The answer is yes, our board was designed specifically for just that, as is the CPU card's memory area.

Tom Jones, 295th Avn Co, SFTS Det., APO N.Y. 09028 is working on a portable speech synthesizer for a handicapped child. Tom would appreciate contact with someone who has experience in this area, and who would be willing to shop for hard to get parts for him in the USA. This is a very worthwhile project in a new frontier area of applied microprocessors.

*** FOR SALE ***

COMPLETE NETRONICS ELF II COMPUTER SYSTEM. ITEMS INDICATED AS UNASSEMBLED ARE NETRONICS SUPPLIED BOARD & PARTS KITS STILL IN ORIGINAL PACKING, DOCUMENTATION INCLUDED. ALL ASSEMBLED ITEMS ARE IN WORKING ORDER. ALL PRICES ARE CANADIAN AND INCLUDE POSTAGE IN CANADA AND USA. ALL ITEMS CAN BE PURCHASED SEPERATELY OR SAVE BY BUYING ENTIRE SYSTEM.

1-	ASSEMBLED NETRONICS ELF II REV D COMPUTER E/W 5 - 86 PIN CONNECTORS C/W RCA-1802 MANUAL MPM-201B TOM PITTMAN'S SHORT COURSE IN PROGRAMMING	80.00
1-	ASSEMBLED VECTOR ALUMINUM CARD CAGE MODIFIED TO HOLD 16 NETRONICS SIZE CARDS C/W NETRONICS 10 - CARD ELF BUS EXPANDER BACKPLANE E/W 10 - 86 PIN CONNECTORS	100.00
1-	UNITRON 'BLACK BEAUTY' HEAVY DUTY SWITCHING POWER SUPPLY +5V @5A, -5V @.5A, +12V @2.5A, -12V @.5A	50.00
1-	WALL MOUNTED 8.5VAC ADAPTER	4.00
1-	LEEDEX BLACK & WHITE 12" VIDEO 100 MONITOR	90.00
1-	NETRONICS IBM BLUE & BLACK STEEL KEYBOARD ENCLOSURE	10.00
1-	ASSEMBLED NETRONICS VIDEO DISPLAY BOARD 24 X 16 CHAR	50.00
1-	ASSEMBLED NETRONICS SMARTVID VIDEO DISPLAY BOARD STANDARD VERSION 80 X 24 CHAR 5 X 7 DOT MATRIX 15.6KHZ	120.00
2-	ASSEMBLED NETRONICS 4K STATIC RAM BOARDS @45.00 EA	90.00
3-	BARE NETRONICS KLUGE (PROTOTYPE) BOARDS @10.00 EA	30.00
1-	UNASSEMBLED NETRONICS GIANT BOARD KIT	25.00
1-	UNASSEMBLED NETRONICS 16K STATIC RAM KIT	110.00
1-	UNASSEMBLED NETRONICS FULL BASIC KIT INCLUDING HARDWARE MATH BOARD C/W FULL BASIC ON 2716 EPROMS (CASSETTE VERSION ALSO INCLUDED)	80.00
1-	UNASSEMBLED NETRONICS EPROM BURNER KIT	30.00
1-	UNASSEMBLED NETRONICS A-D/D-A BOARD KIT	25.00
1-	UNASSEMBLED NETRONICS ELECTRIC MOUTH KIT C/W DIGITALKER VOX I & II WORD SETS	70.00
1-	PARTIALLY ASSEMBLED COLOR GRAPHICS/MUSIC KIT	30.00
1-	NETRONICS LIGHT PEN KIT	5.00
1-	MODIFIED RADIO SHACK MODEL CTR-41 CASSETTE RECORDER	25.00
1-	MODIFIED RADIO SHACK MODEL CTR-40B CASSETTE RECORDER	25.00
1-	ACE BARE 8" DISC CONTROLLER BOARD & ACE BARE DMA ADAPTER BOARD (ELF II)	10.00
1-	ACE FORTH NETRONICS CASSETTE FORMAT (6K)	5.00
1-	'DEFACTO' YEAR 1-6	10.00
1-	NETRONICS PILOT LANGUAGE ON CASSETTE	10.00
1-	NETRONICS ELFBUG MONITOR ON CASSETTE	10.00
1-	NETRONICS ASSEMBLER & DISASSEMBLER ON CASSETTE	10.00
1-	NETRONICS TEXT EDITOR ON CASSETTE	10.00
1-	NETRONICS TINY BASIC ON CASSETTE C/W TOM PITTMAN'S SHORT COURSE ON TINY BASIC	10.00
	TOTAL	1124.00
	ENTIRE SYSTEM FOR	925.00

ROGER FLIEGEL
1520 RANCHLANDS ROAD N.W., CALGARY, ALBERTA, CANADA, T3G 1N1
PHONE 403-239-1251

REVIEW - PASCAL WITH YOUR BASIC MICRO

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

Pascal With Your BASIC Micro
Jeremy Ruston
Howard W. Sams & Co., Inc.
Indianapolis, IN: 1983
$9.95

Many colleges require Pascal as a prerequisite to more advanced software classes. Many software textbooks present example programs in Pascal (or the very similar Algol or PL/M). However, the student who wants to study in more depth than provided for by the over crowded computer lab schedule and even more the person studying independently will likely be frustrated by the lack of inexpensive Pascal compilers for the microcomputers available to him/her.

Jeremy Ruston has met this need in his book, PASCAL ON YOUR BASIC MICRO. In the book he provides a listing in BASIC for a cross compiler from Pascal to BASIC.

In order to use Ruston's compiler a mass storage device, a BASIC with array and string manipulations (DIM, MID$, and string concatenation) and 16K of free memory are needed. Cassette tape can be used for mass storage. Quest BASIC will work. Tiny BASIC would require simulating the array and string manipulations in machine language calls and because it doesn't tokenize will require more memory. Getting the compiler into 16K requires that all remarks be deleted (many of Ruston's remarks state the obvious) and error messages shortened to numbers. Just how compact the compiler becomes depends on how efficient your BASIC's tokenizing is. The cross compiler can be shortened an additional ten to fifteen percent by collecting repeated code into subroutines.

The bad news is: 1) Only a subset of Pascal is supported. 2) No code optimization is performed. 3) Compilation is of glacial speed. I clocked it at 4 1/3 lines per minute on complex expressions. and 4) Functions and procedures (Pascal's names for subroutines) must be hand linked.

The good news is: 1) Ruston's Pascal has thorough error detection. 2) Unlike many tiny Pascals, Ruston's supports variables of type REAL (assuming your BASIC supports floating point arithmetic). 3) Because it is fully disclosed it can be used as an example of compiler implementation. 4) It can be modified to compile subsets of similar languages. Algol, FORTRAN, and PL/M should be easy and Forth and LISP very difficult.

When using the compiler to learn Pascal, I suggest that the student work through the exercises in the first six chapters of Ruston's book before starting a formal class. If a printer is available the completed exercises can be listed (by modifying lines 1350, 1380, 1530, and 1550 of the compiler) and used as examples of working programs. During formal study a student can

REVIEW - PASCAL WITH YOUR BASIC MICRO (cont.)

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

try out his/her design before jousting for a terminal in the computer lab. Where I studied Pascal computer lab sardining was bad enough that the instructor suggested that we use any Pascal compiler available to us.

Another use for Ruston's compiler is to check out algorythms in a structured environment before coding them in assembler. Use of this technique can produce code so polished you can see yourself in it.

As an example of what Ruston's compiler does let's take the routine called <ENGINEERING SPACE/WARP ENGINES SUBMODULE (4)> from STAR SHIP SIMULATIOM-1. I have modified the routine to be a stand-alone program which can be used for determining scaling (fudge) values. The Pascal "source" code is easily written by referring to Garrett's sketch code on page 96 and his list of variables on pages 48 and 49. See the Pascal "source" listing which was entered into Ruston's compiler and the BASIC "object" listing which was automatically generated as compiler output. Pascal requires variables to be declared to be of some type (INTEGER and REAL in this case) before they are used. In Pascal (and other structured languages) the IF...THEN construct is used only to determine whether a statement list will be executed and is not used in completing loops. Compare the next to last line in the Pascal listing with line 260 in the BASIC "object" code. FOR, REPEAT, UNTIL, and WHILE are used for loop control. This is one of the factors which improves the readability of structured languages. Ruston's compiler creates constants called TRUE and FALSE which appear in the BASIC listing but are not used in the example program. Pascal teachers usually require meaningful variable names which must be hand shortened for most BASIC's. The remainder of the Pascal listing can be understood by comparing it to the generated BASIC listing.

For those of us who obtained an 1802 based microcomputer for educational use, a Pascal compiler is a logical addition.

1. Garrett, Roger. STAR SHIP SIMULATION. dilithium Press. Portland, OR: 1978

PASCAL:

```
VAR
  CH, CI, CJ, MOD: INTEGER;
  F2, CK, CU: REAL;
BEGIN
  F2 := 32000;
  CH := 50;
  REPEAT
    WRITE ('ENTER WARP FACTOR');
    READLN (CI);
    CU := 3*CI*CI;
    WRITELN ('VELOCITY = ',CU,' HUNDRED THOUSAND KM/S.');
    WRITE ('ENTER RELIABILITY MODIFIER');
    READLN (MOD);
    IF (CI>0) AND (CH>0)
      THEN BEGIN
      F2 := F2-CU/CH;
      CH := CH+MOD;
      END
    WRITELN ('ENERGY = ',F2);
    WRITELN ('FUNCTIONAL STATUS = ',CH);
    WRITELN ('')
  UNTIL CU<=0;
END.
```

BASIC:

```
 10  REM --CONST--
 20  TRUE=-1
 30  FALSE=0
 40  REM --BLOCK--
 50  REM --BEGIN--
 60  F2=32000
 70  CH=50
 80  REM REPEAT
 90  PRINT "ENTER WARP FACTOR";
 100  INPUT CI
 110  CU=3*CI*CI
 120  PRINT "VELOCITY = ",CU," HUNDRED THOUSAND KM/S."
 130  PRINT "ENTER RELIABILITY MODIFIER";
 140  INPUT MOD
 150  IF (CI>0)(CH>0) THEN  180
 160  GOTO  230
 170  REM IF
 180  REM THEN
 190  REM BEGIN
 200  F2=F2-CU/CH
 210  CH=CH+MOD
 220  REM END
 230  PRINT "ENERGY = ",F2
 240  PRINT "FUNCTIONAL STATUS = ",CH
 250  PRINT ""
 260  REM UNTIL
 270  IF NOT(CU<=0) THEN  80
 280  END
```

ACE HARDWARE - PLEASURES, PROBLEMS AND SUGGESTIONS

Don Stewart, 19983 39A Ave., Langley, B.C., V3A-7G3.

I feel as though I am probably the last person to get my ACE system up and running - I have had some boards finished for over a year now, but, except for the memory, no convienient way to test them until I got it all together. 'All' is the front panel, back-plane, CPU, and 64K RAM. The 6847 Video, version II will follow - hopefully it will go more quickly than some of the others.

I did have some difficulty getting my ACE system to work. Probably I learned far more about the system and the 1802 than I would have had it started easily, but then there were evenings when I felt I was learning too much. This article will describe the areas of difficulty I had - perhaps I'm not actually the last person.

(a) Back-plane (ver 2) - This went together quite well for me. I did have two shorts in the buss which the notes caution about. The 4050 buffer in the cassette circuit has outputs 2, 4, and 6 tied Hi rather than the inputs - 3, 5, and 7. I also found that the 2N3904 was not quite able to drive my current loop terminal - I substituted a 2N4355 from my ELF II. Otherwise the backplane was great.

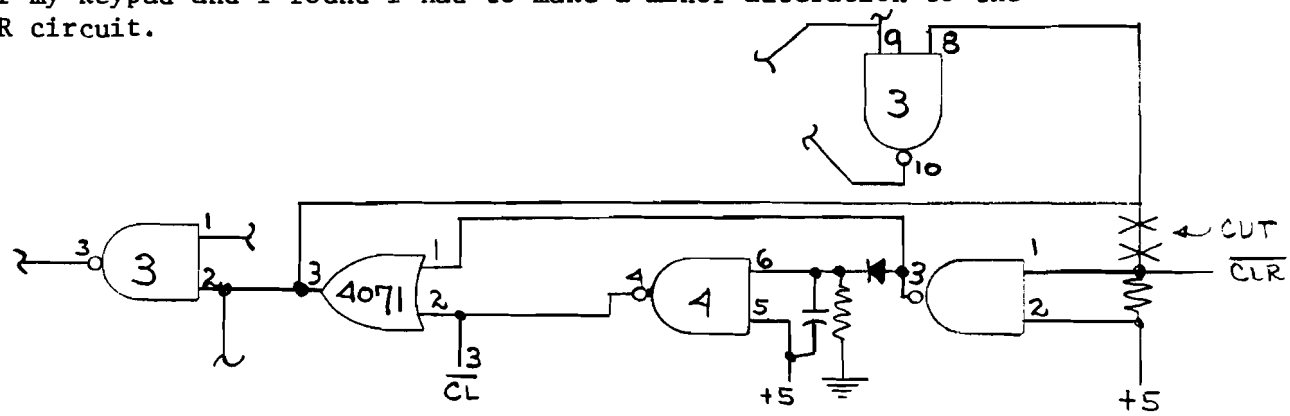
(b) Front Panel - This is really where I had my difficulties and yet, in retrospect, the problems were not that bad. The artwork was excellent and most of my confusion just came from the drawings. Resistor and capacitor values on the layout would have helped but you do pay more attention without them. Most of the time went to keying in, over and over again, the tape load program - I almost got it memorized. As to the drawings: the Reset switch operates $\overline{CLR}$, the Load switch $\overline{WAIT}$, the LEDs for the EPROM burner go to +5 VDC, not common, and the page and 2K LEDs should be driven by a buffer (I had to disconnect them to get the burner to work reliably), D8 is shown backwards and D6 and D7 designators are reversed on the schematic. The switch numbers do not agree between the layout and schematic.

Thank you, Mike, for your suggestion to locate the hex keypad separate from the front panel. I am very pleased to have it on a flat cable, easy to operate. Now that I have the ACE Monitor on EPROM, I don't seem to use the keypad, but I sure did, very often, while trying to get started. The area where the keypad should go I plan to use for Ni-Cad batteries for the clock. Finally, the crystal I bought turned out to be the low-power type mentioned in the clock data sheet and indeed a 200K resistor is required between crystal and clock.

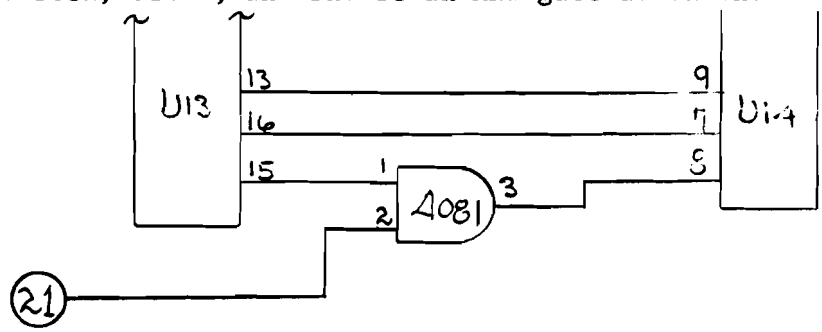
The front panel, operational, is truly neat - I'm very pleased with the hexadecimal displays and especially delighted to be able to program EPROMs so readily (I did install a ZIF socket).

(c) CPU Board - My confusion on the ACE buss over the reset and load switches continued here. I was trying to get DMAIN to work

for my keypad and I found I had to make a minor alteration to the CLR circuit.



(d) RAM Board - I had this board operational on my ELF II with a hay-wired interface - it had gone together very well except for my mistakes. (My mistakes took up most of my time on all of the boards) I did make a minor change; since I already had all 64K populated, I chose to disable the memory here for the clock's memory mapping. The revision is simple; cut the trace from U13-15 to the dip switch, U14-8, and insert an AND gate as shown.



All in all, I quite enjoyed stuffing the boards and getting the system to work. My sincere thanks to the people who did the real work putting the system together.

ELF II INPUT KEY DEBOUNCE

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

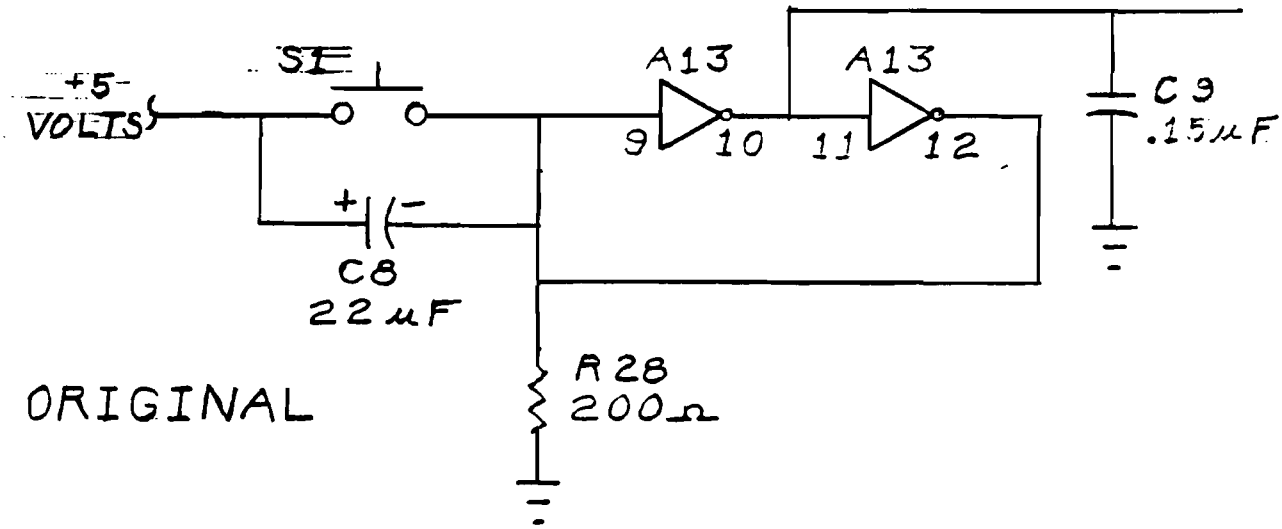
Upon completing my ELF-II, I started Tom Pittman's machine language course. By the time I was writing programs over 30 bytes long, a rapidly worsening attack of byte doubling was evident. I tried lubricating the input key, S1. This helped, but loading an eighth-K program still took two or three tries. Upon inspection of the input switch debounce circuit (see the first figure), I found some things I wanted to change: 1) C8 was connected directly across S1. When C8 was charged to 5 volts and S1 closed, the initial current flowing through S1 was limited by the possibly less than one ohm of wiring resistance. Five amps, even very briefly, is over ten times the current at which most key switches are designed to operate, and can lead to rapid aging of the switch. 2) C9 slows transitions at pin #10 of A13. The longer the input of a logic device stays close to its threshold, the greater the chance of picking up a stray noise pulse. 3) When pin #12 of A13 is low and S1 is then pressed, the current flowing through S1, then into ground via A13, is primarily limited by the on resistance of the FETS in A13. This could produce ground noise.

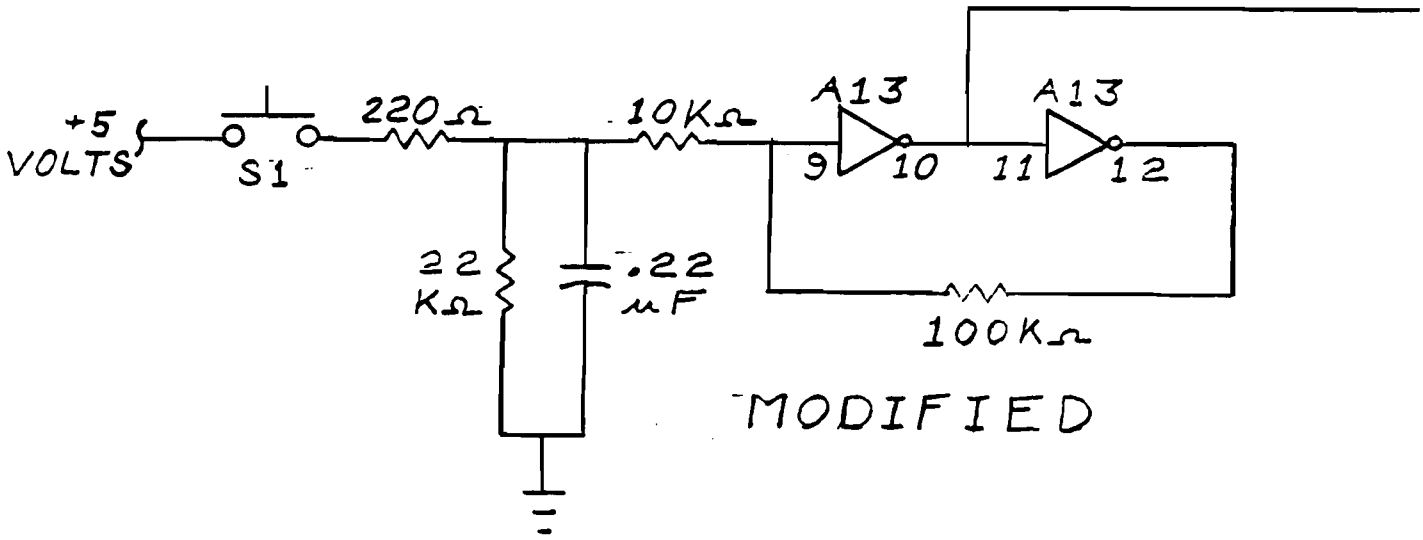
To provide more predictable currents, the circuit in the second figure was constructed. The switch and capacitor current is limited by the 220 ohm resistor. The 22K ohm resistor and the 0.22 uF capacitor have the same time constant as the old C8, R28 combination, but allow the addition of the 220 ohm resistor without major impact on the time constant. The 10K ohm resistor and the 100K ohm resistor combine to provide hysterisis which is nearly independent of A13's characteristics. Besides the parts shown in the diagram, a 16-pin IC socket and either a 16-pin or an 18-pin dual inline component header are needed. If an 18-pin header is used, the two extra pins are cut off and the lug above one of these pins will be used to support the node where the capacitor, the 220 ohm resistor, the 22K ohm resistor, and the 10K ohm resistor join.

Bend pins 9 and 12 of the socket outward. Solder the remaining pins to the corresponding header pins. When connecting the remaining components, position them so that when the header is installed in A13's original socket, the new components will not contact components on the circuit board. Connect the 100K ohm resistor between pins 9 and 12 of the socket. Connect the 0.22 uF capacitor "piggyback" across the 22K ohm resistor. Connect one end of this "piggyback" pair to lug #8 of the header. Connect one end of the 220 ohm resistor to lug #9 of the header. Connect one end of the 10K ohm resistor to the lead of the 100K ohm resistor which connects to pin #9 of the socket. Connect the three remaining resistor leads together (one each from the 22K ohm resistor, the 220 ohm resistor, and the 10K ohm resistor). Lug #12 of the header should have no connection. Check the construction against the diagram of the modified circuit. Remove C8, C9, and R28 from the circuit board. Remove A13 from its socket and insert it into the socket which is mounted on the header. Plug the completed assembly into the A13 socket on the circuit board.

Check the circuit by loading and inspecting a program.

This circuit has been 100% effective for me and should work wherever debouncing a single pole, single throw switch is desirable.





FORTH RECURSION: AN UPDATE

by Alberto Magnani
Home: 22-68 28 Street, Astoria, N.Y. 11105 (212) 545-3675
School: 410 Memorial Drive, Cambridge, MA. 02139 (617) 225-8251

I finally received my Forth manuals and found them to be extremely useful. As a matter of fact, I found a much more elegant way of implementing recursion. We can simply stop compilation, use the smudge bit to let forth know that the definition there but incomplete, then use the word itself in the definition (it will find it in the dictionary), finish the definition, and finally flip the smudge bit again. For example, the factorial word can easily be defined like this:,

```
: N! [ SMUDGE ] DUP 0= IF DROP 1 ELSE DUP 1 - N! * THEN ; SMUDGE
```

Or FIB as:

```
: FIB [ SMUDGE ] DUP 1 = IF DROP ELSE
 DUP 2 = IF DROP 1 ELSE DUP 1 - FIB
 SWAP 2 - FIB + THEN THEN ;  SMUDGE
```

That's all there is to it. I also found that in FORTH Dimensions, they had defined a word, MYSELF, which accomplishes the same (this was in volume 4, which I hope is in the mail). Also in volume 5, they solve another recursion problem, that of forward references. i.e. - One word calls a second word which in turn calls the first. They use variables and vectored execution to solve this problem very elegantly.

In any case, I am back at school now, with little time left before registration day, so I will not be able to do too much until the summer. But I did start writing up the code for a LISP interpreter. The biggest problem I've run into is figuring out exactly what must get compiled first. Right now I have several sheets of code in haphazard order which has to be straightened out ... wishful thinking would be nice in forth. Hopefully, I'll have something reasonable put together by the end of summer. This summer I also hope to get a disk system working, an EPROM burner and additional memory ... the struggle for a good system continues. One last note, I did find that the book I referred to in my last letter will indeed be released in June, by the MIT Press (co-published by McGraw Hill), in case anyone is interested. It is an excellent book.

CALENDAR MAKER

Bart Mathias 4894-3 Kilauea Ave. Honolulu HI 96816 USA

I bet there are a lot out here like me, and that's why there's danger of a cover-only *Ipso Facto*. I have had a variety of programs that I want to send ever since becoming a member in 1981. But there's always one little place that is particularly inelegant to fix before sending in each program, and they run with the 1861 and meanwhile you've had to add an 80x24 monitor to do the part-time work that provides the money for your hobby, because an irreplaceable high-voltage capacitor went out in the TV you had converted to a monitor and you spend so much time using your computer on the job that you don't have time to finish the job and make it convertible while your FORTH tape sits untried for over a year and your 64K board, your disk controller board and all the components sit unsoldered gathering dust on your work table for the same length of time. Isn't that why you other guys haven't sent anything in yet either?

Well, actually none of my programs are that hot, and that's why I planned to send a bunch and let the Editors see if there were any worth printing. But I was determined to at least offer this one for a fall issue. It's now 21 Jan; well, there's always fall 1984.

This might be a useful program for people who like to have some calendars with a little room to jot notes of things done or to do in the date rectangles. It is written for an ELF II and an Epson MX80 (original, Graftrax80, or Gratrax^{PLUS} versions), but I imagine it can be easily converted (I have even written in CR's before the LF's--redundant for Epsons--just in case). The user inputs the year number, in ASCII, e.g. 31 39 38 34, via the hex keypad, followed by the length of February that year in hex: 1D if it is leap year, 1C otherwise; followed finally by the sequential number (Sunday = 01, Saturday = 07) of the first day of the year. The program will automatically reduce the vertical height of the rectangles in months with six instead of five calendar weeks so it can fit on a page without overlapping.

The program loads in any two consecutive pages. The order of the two blocks may be reversed if desired; the only change necessary is to make FC in the third byte of the currently first block FF instead.

Register usage: [in brackets: initialization phase only]

- R0: PC
- R2: STAK
- R6.1: last date of month (LOM)
- R6.0: current month day (DOM)
- R7.0: DOM (-10 (-10 (-10))) during ASCII conversion
- R8.1: first weekday of current week (FOW)
- R8.0: counter for horizontal and vertical lines

```
R9:  year-number address (YEAR)/weekday name list addr.
RA:  [HT list address] next month name, length address
RD:  print routine address
RE:  [February length address]
  RE.0:  weekday counter (DOW)
RF.1: lines per week (VRTCL)
RF.0: counts tens in month day number (TENS)

                                  Initialization
xi 00: 90 B2                STAK this page (could be next page)
   02: FC 01                data lists and print routine
   04: B9 BA BD BE               are on next page
   08: F8 FD A2             STAK address
   0B: F8 99 A9             YEAR initialization address
   0E: F8 0B AA             HT list address
   11: F8 01 AD             print routine address
   14: F8 2B AE             February length address
   17: E2                   SEX R2
   18: 4A DD                send HT list
   1A: 3A 18                     to printer
   1C: E9                   SEX YEAR
   1D: 3F 1D                wait for
   1F: 37 1F                     INPUT key
   21: 6C 64                then read and display hex
   23: 89 FF 9D                  until the four digits of
   26: 3A 1D                     the year are filled in
   28: E2                   SEX back to R2
   29: 3F 29                wait for next input
   2B: 6C 5E                read and store length of February
   2D: 64 22                verify correct keystrokes
   2F: 37 2F 3F 31          wait for one more input
   33: 6C 64 22             read, display first weekday of year
   36: 38                   jump with it
                       Start monthly loop
   37: 8E                   get current DOW
   38: FA 07                make sure it's no more than SAT
   3A: B8                   and put in FOM
   3B: 8A FF 92             check for and handle
   3E: 33 (3E)                   end of year (e.g., stop)
   40: F8 0A DD DD DD       start page with 3 x LF
   45: 4A DD                write name
   47: 3A 45                     of the month
   49: 4A B6                load, set LOM
   4B: 52 98 F4             add to FOM and subtract
   4E: FF 25                     37 to see if 5 or 6 week month
   50: CF                        setting VRTCL to
   51: F8 0A                     10
   53: C7                        or
   54: F8 08                     8
   56: BF                        respectively
   57: F8 09 DD DD DD DD DD then HT across the page
   5E: F8 96 A9             set R9 to YEAR
   61: 49 DD                     and print
   63: 3A 61                     it
   65: A6                   reset DOM to 00
```

```
xi 66: 49 DD                 print weekday
   68: 3A 66                      names
                        Start weekly loop
   6A: F8 0D DD                  CR (no LF)
   6D: F8 50 A8                  set counter to 80 (char/line)
   70: 28                        print a
   71: F8 5F DD                       row of _
   74: 88
   75: 3A 70
   77: 86 52                     End of month? (LOM
   79: 96 F7                          - DOM
   7B: 3A 83                          = 0?)
   7D: F8 0C DD                  If so, new page,
   80: 1E                             next DOW,
   81: 30 37                          and go do new month
   83: F8 00 AE                  else reset DOW
   84: F8 0D DD F8 0A DD CR, LF
                        Start daily loop
   8C: 1E                        next DOW
   8D: 98 52                     first day this calendar week?
   8F: 8E F7                          (DOW - FOW
   91: 3B C3                          < 0?) not yet?
   93: 3A 9A                          = 0?) past the first?
   95: F8 7C DD                       > 0?) just?  then type !
   98: 30 9D                          and jump
   9A: F8 20 DD                  past, so no left outside border
   9D: F8 7C DD                  and type inside border
   A0: 16                        next DOM
   A1: 88                        VRTCL
   A2: 3A C3                          = 0?
   A4: F8 0E DD                  if so, expand print
   A7: F8 00 AF                       and reset TENS, to
   AA: 86                             convert DOM
   AB: A7                             to
   AC: FF 0A                          ASCII
   AE: 3B B3                          and
   B0: 1F                             print
   B1: 30 AB                          it
   B3: 8F                             out
   B4: 3A B9                     with, if under 10,
   B6: F8 20                          a SP in 10's place
   B8: C8                             ...
   B9: FC 30
   BB: DD
   BC: 87 FC 30 DD
   C0: F8 14 DD                  back to 10 char/inch
   C3: F8 09 DD                  HT to righthand side of day space
   C6: 86 52                     end of month?
   C8: 96 F7                          (LOM - DOM
   CA: 32 D1                          = 0?)
   CC: 8E FF 07                  if not, then SAT done?  (DOW - 7
   CF: 3A 8C                          = 0?)
   D1: F8 20 DD                  if either, SP to right border pos
   D4: F8 7C DD DD                    and type !!
   D8: 88                        counter = 0 (first line of week)?
```

```
xi D9: 3A E7                if not, branch
   DB: 9F A8                otherwise, set counter = VRTCL
   DD: 8E 52                DOW to STAK
   DF: B6 F7                 and subtract it from DOM,
   E1: CF                     if result not less than 00
   E2: F8 00                    (in which case set up to 00)
   E4: A6                     reset DOM for another horiz. pass
   E5: 30 83                and go back and make it
   E7: 28 88                if past first line, count one off
   E9: 3A DD                and if not finished, go cross again
   EB: F8 01 B8             else, set FOW to 1 (Sunday)
   EE: 30 6A                and go start new week

                        Subroutine and Data
xj 00: D0                 Your printer routine goes here. I
   01: 22 52              said I've been busy. I haven't had
   03: 3E 03              time to replace my oh-boy-let's-try-
   05: 7B                 the-new-printer emergency software
   06: 67                 interface with hardware for 2 1/2
   07: 7A 7B              years!
   09: 30 00
   0B: 1B 44                ESC D introduces HT list to Epson,
   0D: 0B 16 21 2C          the list,
   11: 37 42 4D 00             with 00 terminator
   15: 0E 4A 41 4E 55 41 52 59 14 00   "JANUARY"
   1F: 1F                                31 (days long)
   20: 0E 46 45 42 52 55 41 52 59 14 00
   2B: xx                   February length will go here
   2C: 0E 4D 41 52 43 48 14 09 00 1F
   36: 0E 41 50 52 49 4C 14 09 00 1E
   40: 0E 4D 41 59 14 09 00 1F          { ^N (0E) expands,
   48: 0E 4A 55 4E 45 14 09 00 1E       { ^T (14) nrmlizes,
   51: 0E 4A 55 4C 59 14 09 00 1F
   5A: 0E 41 55 47 55 53 54 14 00 1F
   64: 0E 53 45 50 54 45 4D 42 45 52 14 00 1E
   71: 0E 4F 43 54 4F 42 45 52 14 00 1F
   7C: 0E 4E 4F 56 45 4D 42 45 52 14 00 1E
   88: 0E 44 45 43 45 4D 42 45 52 14 00 1F
   94: 07                   BEL (clapper removed)
   95: xx
   96: 0E 20 20 xx xx xx xx  "    1984(?)"
   9D: 0A 0A 00             LF twice after month and year
   A0: 0E 20 53 75 6E 14 09          "  Sun     "
   A7: 0E 20 4D 6F 6E 14 09 0E 20 54 75 65 14 09
   B5: 0E 20 57 65 64 14 09 0E 20 54 68 75 14 09
   C3: 0E 20 46 72 69 14 09          "  Fri     "
   CA: 0E 20 53 61 74 14 00          "  Sat     "
```

NOVEMBER 1984

Sun	Mon	Tue	Wed	Thu	Fri	Sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

CALENDAR MAKER

Absolute tested and final version, 22 January 1984

		_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
05	00:	90	B2	FC	01	B9	BA	BD	BE	F8	FD	A2	F8	99	A9	F8	0B
	10:	AA	F8	01	AD	F8	2B	AE	E2	4A	DD	3A	18	E9	3F	1D	37
	20:	1F	6C	64	89	FF	9D	3A	1D	E2	3F	29	6C	5E	64	22	37
	30:	2F	3F	31	6C	64	22	38	8E	FA	07	B8	8A	FF	92	33	3E
	40:	F8	0A	DD	DD	DD	4A	DD	3A	45	4A	B6	52	98	F4	FF	25
	50:	CF	F8	0A	C7	F8	08	BF	F8	09	DD	DD	DD	DD	DD	F8	96
	60:	A9	49	DD	3A	61	A6	49	DD	3A	66	F8	0D	DD	F8	50	A8
	70:	28	F8	5F	DD	88	3A	70	86	52	96	F7	3A	83	F8	0C	DD
	80:	1E	30	37	F8	00	AE	F8	0D	DD	F8	0A	DD	1E	98	52	8E
	90:	F7	3B	C3	3A	9A	F8	7C	DD	30	9D	F8	20	DD	F8	7C	DD
	A0:	16	88	3A	C3	F8	0E	DD	F8	00	AF	86	A7	FF	0A	3B	B3
	B0:	1F	30	AB	8F	3A	B9	F8	20	C8	FC	30	DD	87	FC	30	DD
	C0:	F8	14	DD	F8	09	DD	86	52	96	F7	32	D1	8E	FF	07	3A
	D0:	8C	F8	20	DD	F8	7C	DD	DD	88	3A	E7	9F	A8	8E	52	86
	E0:	F7	CF	F8	00	A6	30	83	28	88	3A	DD	F8	01	B8	30	6A
	F0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	0C	1F	FF	FF
06	00:	D0	22	52	3E	03	7B	67	7A	7B	30	00	1B	44	0B	16	21
	10:	2C	37	42	4D	00	8E	4A	41	4E	55	41	52	59	14	00	1F
	20:	0E	46	45	42	52	55	41	52	59	14	00	1D	0E	4D	41	52
	30:	43	48	14	09	00	1F	0E	41	50	52	49	4C	14	09	00	1E
	40:	0E	4D	41	59	14	09	00	1F	0E	4A	55	4E	45	14	09	00
	50:	1E	0E	4A	55	4C	59	14	09	00	1F	0E	41	55	47	55	53
	60:	54	14	00	1F	0E	53	45	50	54	45	4D	42	45	52	14	00
	70:	1E	0E	4F	43	54	4F	42	45	52	14	00	1F	0E	4E	4F	56
	80:	45	4D	42	45	52	14	00	1E	0E	44	45	43	45	4D	42	45
	90:	52	14	00	1F	07	0E	0E	20	20	31	39	38	34	0A	0A	00
	A0:	0E	20	53	75	6E	14	09	0E	20	4D	6F	6E	14	09	0E	20
	B0:	54	75	65	14	09	0E	20	57	65	64	14	09	0E	20	54	68
	C0:	75	14	09	0E	20	46	72	69	14	09	0E	20	53	61	74	14
	D0:	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
	E0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
	F0:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
07																	

THE ELF TELLS TIME
(A TALKING ELF CLOCK)

By Bob Briggs, Los Altos, CA

Does your ELF computer spend time collecting dust in the corner of your workbench? Give your ELF a full time job --- announcing the time of day for you. This article describes a talking clock using the Netronics ELF II microcomputer with the National Semiconductor DT1050 DIGITALKER chip set. The ELF clock will automatically announce the time on the hour, or at any time when requested, with a statement such as "THE TIME IS TWO EIGHTEEN PM." Although not as elegant as a grandfather clock with its stately chimes and expensive furniture decor, this ELF clock can be a useful timekeeper and conversation piece for you. Or, if you simply want details on connecting DIGITALKER to the RCA 1802 microprocessor (on the ELF II bus), that's here too.

We begin with a brief description of DIGITALKER, then describe interface circuitry, checkout software, and the clock program, and conclude with operating instructions for the talking clock.

The DIGITALKER DT1050 chip set comprises a Speech Processor Chip (SPC) and two 64K vocabulary ROMs. Together these provide 144 spoken words, letters, numbers, beeps and pauses. (See Table for vocabulary list and associated word codes.) It takes 55 seconds to speak the entire vocabulary in rapid succession, which indicates that the bit rate has been reduced by a factor of more than 25 compared to that used for digital transmission of telephone speech. Yet DIGITALKER intelligibility is extremely good. For more details on DIGITALKER, see "Build a Low-Cost Speech-Synthesizer Interface", BYTE, June 1981.

DIGITALKER is easily interfaced to any eight bit microprocessor bus. As shown in the block diagram of Figure 1 the microprocessor need only be connected to the SPC, which in turn handles all communications with the vocabulary ROMs. The microprocessor controls the SPC by placing a word code on the data bus and strobing (pulsing) a control line. DIGITALKER then outputs the word on the audio line and, when finished speaking, raises the interrupt line to tell the microprocessor that it is ready for another word. In comparison with other popular microprocessors, the ELF CPU (1802) is ideal for this type of controller application because it provides all required interface signals without the necessity for separate input/output (I/O) support chips.

CIRCUIT DESCRIPTION

The actual circuit diagram is similar to the block diagram and is shown in Figure 2. Functions on the left side of the circuit diagram connect directly to the 1802 microprocessor via the ELF II bus at the pin numbers indicated. Speech is initiated by strobing low the WRITE BAR line. In this design the strobe is generated by using the 1802 N1 line, which corresponds to the OUT

2 (62) instruction. The interrupt signal from DIGITALKER (which tells when the SPC is ready to receive another word command) is connected to External Flag 2 and is thus accessible with the B2 and BN2 instructions. Although only one vocabulary ROM is shown there are two ROMs connected in parallel. The pin names on the two ROMs are identical except for pin 20, which is CS on one ROM and CS BAR on the other. The audio output from the SPC drives an amplifier, filter, and speaker.

Although the circuit is straightforward, some items encountered during its construction are of interest.

I wire wrapped the circuit on a Netronics kluge board, which plugs into the main ELF II circuit board. If you don't have the kludge board you may be able to rig something up since there are only about 16 bus connections used.

The SPC and output amplifier require a filtered DC supply of 7 to 11 volts. DIGITALKER current draw from this supply is about 55 milliamperes while talking and 45 milliamperes quiescent. I tried using the unfiltered eight volt supply (which feeds the five volt regulator) as a voltage source for the SPC but this resulted in excessive hum in the audio output. A separate filtered supply or a nine volt transistor radio battery will work fine.

The SPC requires a 4 Mhz crystal , which I found at a local electronics store. I had considered using the 3.58 Mhz crystal that drives the 1802 microprocessor but the SPC clock input requires more than a five volt swing, and the DIGITALKER specification sheet calls for a frequency tolerance of plus or minus two percent.

Since the Netronics kluge board provides room for only two oversize sockets, and DIGITALKER requires three, I soldered the two vocabulary ROMs piggy-back and plugged them into one 24 pin socket. (If you find this necessary use a low wattage iron with a pencil lead thin tip.) The 40 pin SPC uses the other oversize socket.

The LM386 output amplifier will supply about one-half watt to an 8 ohm speaker, which is enough for comfortable listening across a room. To prevent oscillation I had to connect a 0.1 uf bypass capacitor across the LM386 supply and ground pins right at the chip.

The entire circuit as shown in Figure 2 (except speaker) takes only a small portion of the kluge board and leaves plenty of room for other projects.

PROGRAMMING

The programs in this article are written for the minimum ELF configuration (256 bytes of memory) and thus when a register is used to point to memory only the low byte is initialized.

The SPC resets its interrupt line whenever it receives a valid word command, and sets the line high at the completion of the spoken word. Thus the program can use the interrupt line from the SPC to tell when DIGITALKER is ready to accept a new word command. If a new word command is sent before the interrupt line goes high the new word will begin immediately and truncate the old one.

Notice that the programs do not test whether the interrupt line is high until after a word code is sent to DIGITALKER. This is because, at power up, DIGITALKER resets the interrupt line and it stays reset until a word is finished. If the program were to wait for the interrupt line to go high before sending the very first word command it would wait forever.

Once the circuit in Figure 2 is ready you will want to check it out with a simple driver program (see Listing 1). When this program is run, DIGITALKER will say "THIS IS DIGITALKER. IT IS OK". You can make it say anything in its vocabulary by changing the code starting at address 13 HEX according to the word list in the Table. Be sure to end the list with the code FF HEX so that the program knows where to stop.

TALKING CLOCK

The TALKING CLOCK program keeps accurate time by counting the 60 Hertz frequency of the 110 volt house current. DIGITALKER will automatically announce the time on the hour with a statement such as "THE TIME IS THREE PM." The program will also announce the time on demand with a statement like "THE TIME IS ELEVEN FIFTY THREE AM." And it lets you sleep by remaining quiet at night.

Before describing the program we need to add a little more hardware to get the 60 Hertz line frequency to the microprocessor. The circuit shown in Figure 3 accomplishes this by coupling the secondary of the power transformer to the ELF interrupt line. A 4N25 opto-isolator provides ground isolation and half-wave-rectification of the AC current. The voltage divider in front of the opto-isolator prevents its three volt reverse voltage limit from being exceeded. False counts due to noise on the house current are prevented by the low pass filter formed by the resistor-capacitor combination in front of the 74C914 schmitt trigger. The clock would mysteriously gain time before I added this filter. A switch allows you to disable interrupts while setting the clock. Other control signals connected to the interrupt line should be disconnected, including that from the 1861 video driver chip on the ELF II. You may wish to use a simpler 60 hertz interface circuit which I noticed after the clock was built. It uses capacitive coupling instead of an opto-isolator and is shown in Figure 1 of the article "CP/M, Your Time Has Come: A Real-Time Clock for the Most Popular Microcomputer Operating System", BYTE, May 1982.

The talking clock program is described by the flowcharts in Figure 4 and by comments in the program listing. An interrupt service routine counts down 3600 cycles (3599 through zero) to form a one minute delay. The main program waits in a tight loop for the I key to be pressed or for one minute to elapse. If the I key is pressed the program goes to the talk routine and announces the time. As each minute goes by the program advances the minute and adjusts the talk code. The time is automatically announced on the hour. AM and PM are interchanged at noon and at midnight. It takes about 45 minutes to key in the code. I recommend battery backup.

CLOCK STARTING AND OPERATING PROCEDURE

For clarity, the following step by step procedure is given for starting and operating the clock.

STEP 1: Disable interrupts with the interrupt switch (see Figure 3) and then load TINY MONITOR (see sidebar) and the TALKING CLOCK program. It is very important to remember to disable interrupts at this point. Otherwise your ELF will persist in wandering off to never-never-land.

STEP 2: If restarting the program (to adjust the time, for example) you must reload instruction FB HEX into address 1E HEX because it is overwritten by the stack anytime an interrupt occurs (i. e., 60 times per second when the program is running with the interrupt switch closed.) As you might imagine, it took me awhile to squeeze this program into the 256 bytes of RAM.

STEP 3: If after 1 AM and before 1 PM use instruction 7A HEX in address 2E HEX; otherwise use 7B HEX. This allows the program to be silent overnight.

STEP 4: Decide the hour and minute that you will start the clock. Load the hour into address 98 HEX and the minute into address 9C HEX. For example, if you plan to start the clock at 11:05 load 11 HEX into address 98 HEX and load 05 HEX into address 9C HEX.

STEP 5: (Optional). This step is necessary only if you want the program to speak correctly (when the I key is pressed) before it automatically adjusts its talk code on the minute and on the hour.

Load the talk code for the hour (as determined from the Table) into address 99 HEX. The program does this automatically on the hour. For example, if the clock is to be started between eleven and twelve o'clock use the talk code 0B HEX (for eleven).

Load the tens talk code into address 9B HEX, and the minute talk code into address 9D HEX. The program does this automatically on the minute. For example, if the clock will be started at 58 minutes past the hour, the correct tens talk code is 17 HEX (for fifty), and the minute talk code is 08 HEX. The tens talk code

should be 2E HEX for minutes Ø1 through Ø9, and code 43 HEX for minutes 1Ø through 19.

STEP 6: Set AM/PM as follows. Load 2Ø HEX into address A1 HEX for AM, or 2F HEX into address A1 HEX for PM.

STEP 7: (Optional.) The program as listed begins announcing the hour at 7 AM, and remains silent after 11 PM is announced. These times can be changed by altering the contents of HEX addresses 79 and 7F.

STEP 8: Run the program by starting execution at location 1E HEX. The hour will be shown on the HEX display. When the preset time arrives, start the clock by closing the interrupt switch. DIGITALKER will announce the time on the hour. If you wish to hear the time immediately, push the I key (see step 5.) When the I key is pressed, the minute is shown on the HEX display while the time is announced.

ANOTHER APPLICATION

Now that I've gotten this far, I decided to throw in another page of code that came in handy once. PROGRAM 2 is a simple timer program which randomly selects one of eight announcements when the preset time has elapsed. It was written to go with the BOGGLE word game by PARKER BROTHERS after it was noticed that the three-minute sandglass timer that came with the game was unreliable. To operate the timer program, first load in the TINY MONITOR (see sidebar), then load PROGRAM 2 and RUN starting at hex address 2Ø. The length of delay is determined by the values of XX, YY, and ZZ as explained in the program comments. For example, for a 3 minute delay use XX = 5D, YY = 6A, and ZZ = 67. Press the I key to start the timer. After ELF/DIGITALKER announces that the time is up the program is ready to go again. Simply press the I key to restart the timer.

IN CLOSING

These programs just begin to demonstrate the applications possible with programmable speech. Try writing some talking programs of your own. Some suggestions are a guessing game in which the program would pick a number for you to guess and announce whether your guess was too low or too high, or a program to teach simple arithmetic with statements like "TWO PLUS TWO EQUALS ?" or "NINE TIMES EIGHT EQUALS ?". When building sentences you might become exasperated at the limited vocabulary. Additional speech ROMS are available for DIGITALKER, or you may prefer one of the Votrax style unlimited vocabulary systems. These latter give lower quality speech, but are improving. The March 1984 BYTE magazine describes one of the latest unlimited vocabulary chips. Also, the price is coming down. When I wrote the TALKING CLOCK code three years ago the price of DIGITALKER was $8Ø, which was more than I paid for my used ELF II. Now it is available for $35 (for example from JAMECO in San Carlos, CA.)

(415) 592-8097

----------SIDEBAR----------

TINY MONITOR PROGRAM

ADDR	CODE	LABEL	MNEMONICS	COMMENTS
ØØ	F8 1D AF		LDI 1D, PLO F	RF=1D
Ø3	EF		SEX F	X=F
Ø4	6C		INP 4	M(RF)=KEYPAD
Ø5	3Ø 1C		BR JMP	GO TO JMP
Ø7	7B		SEQ	Q LIGHT ON
Ø8	3F Ø8	W1	BN4 W1	WAIT FOR I PRESS
ØA	6C AE		INP 4, PLO E	RE=KEYPAD
ØC	37 ØC	NEXT	B4 NEXT	WAIT FOR I RELEASE
ØE	8E		GLO E	GET RE
ØF	5F EF		STR F, SEX F	STORE IN M(RF), X=F
11	64 2F		OUT 4, DEC F	DISPLAY KEYPAD
13	3F 13	W2	BN4 W2	WAIT FOR I PRESS
15	EE		SEX E	X=E
16	C5 C4		LSNQ, NOP	LONG SKIP IF Q=Ø
18	6C		INP 4	STORE KYBD IN M(RE)
19	64		OUT 4	DISPLAY M(RE), RE+1
1A	3Ø ØC		BR NEXT	GO TO NEXT LOC
1C	3Ø 3Ø	JMP	BR ADDR	GO TO M(1D)

An auxiliary program, called an operating system, or monitor, is useful when working with larger programs to allow memory contents to be examined and changed and to allow execution to begin at an arbitrary address. The TINY MONITOR program is provided as an aid to using the TALKING CLOCK program. It has capabilities similar to the "ETOPS-256" operating system in the March 1977 issue of POPULAR ELECTRONICS, but in addition displays the memory address and is a little shorter.

Directions for using TINY MONITOR:

Turn off the interrupt switch (see Figure 3). Load the program and then turn off the RUN, LOAD, and MEMORY PROTECT switches.

To start program execution at any address, key in the address on the HEX keyboard and turn on the RUN switch. The program will run with RØ as the program counter.

To examine memory, start execution at address Ø8 HEX. The Q light will remain off. Key in the address to be examined, then press and release the I key. The address will be displayed. To see

the memory contents, press and hold down the I key. When the I key is released, the next address will be displayed, and its contents can be examined by again pressing and holding the I key.

To change memory contents, start execution at address Ø7 HEX. The Q light will turn on. Key in the address to be changed, then press and release the I key. The address will be displayed. Key in the new memory contents, then press and hold the I key. The new contents will be displayed. When the I key is released the next address will be displayed, and its contents can be changed by keying in the new data and pressing the I key.

To exit the monitor, turn off the RUN switch.

----------END SIDEBAR----------

LISTING 1

INTERFACE CIRCUIT CHECKOUT PROGRAM

ADDR	CODE	LABEL	MNEMONICS	COMMENTS
ØØ	F8 13 AD		LDI START PLO D	RD POINTS TO START
Ø3	ED		SEX D	X=D
Ø4	7B		SEQ	Q LIGHT ON
Ø5	FØ	GTWD	LDX	M(RD) TO D
Ø6	FD FF		SDI FF	D=FF-D
Ø8	32 11		BZ STOP	BRANCH IF LAST WORD
ØA	64 2D		OUT 4 DEC D	DISPLAY WORD CODE
ØC	62		OUT 2	SPEAK WORD, R(X)+1
ØD	35 ØD	WAIT	B2 WAIT	WAIT FOR DIGITALKER
ØF	3Ø Ø5		BR GTWD	NEXT WORD
11	7A	STOP	REQ	Q LIGHT OFF
12	ØØ		IDL	STOP
13	ØØ 47	START	WORD LIST (LAST WORD FOLLOWED BY FF)	
15	61 6Ø			
17	2E 2A			
19	FF			

TABLE SHOWING DIGITALKER WORD LIST

WORD	HEX CODE	WORD	HEX CODE
THIS IS DIGITALKER	00	CENTI	48
ONE	01	CHECK	49
TWO	02	COMMA	4A
THREE	03	CONTROL	4B
FOUR	04	DANGER	4C
FIVE	05	DEGREE	4D
SIX	06	DOLLAR	4E
SEVEN	07	DOWN	4F
EIGHT	08	EQUAL	50
NINE	09	ERROR	51
TEN	0A	FEET	52
ELEVEN	0B	FLOW	53
TWELVE	0C	FUEL	54
THIRTEEN	0D	GALLON	55
FOURTEEN	0E	GO	56
FIFTEEN	0F	GRAM	57
SIXTEEN	10	GREAT	58
SEVENTEEN	11	GREATER	59
EIGHTEEN	12	HAVE	5A
NINETEEN	13	HIGH	5B
TWENTY	14	HIGHER	5C
THIRTY	15	HOUR	5D
FORTY	16	IN	5E
FIFTY	17	INCHES	5F
SIXTY	18	IS	60
SEVENTY	19	IT	61
EIGHTY	1A	KILO	62
NINETY	1B	LEFT	63
HUNDRED	1C	LESS	64
THOUSAND	1D	LESSER	65
MILLION	1E	LIMIT	66
ZERO	1F	LOW	67
A	20	LOWER	68
B	21	MARK	69
C	22	METER	6A
D	23	MILE	6B
E	24	MILLI	6C
F	25	MINUS	6D
G	26	MINUTE	6E
H	27	NEAR	6F
I	28	NUMBER	70
J	29	OF	71
K	2A	OFF	72
L	2B	ON	73
M	2C	OUT	74
N	2D	OVER	75
O	2E	PARENTHESIS	76
P	2F	PERCENT	77
Q	30	PLEASE	78
R	31	PLUS	79
S	32	POINT	7A
T	33	POUND	7B
U	34	PULSES	7C
V	35	RATE	7D
W	36	RE (PREFIX)	7E
X	37	READY	7F
Y	38	RIGHT	80
Z	39	SS (PLURALIZER)	81
AGAIN	3A	SECOND	82
AMPERE	3B	SET	83
AND	3C	SPACE	84
AT	3D	SPEED	85
CANCEL	3E	STAR	86
CASE	3F	START	87
CENT	40	STOP	88
400 HERTZ TONE	41	THAN	89
800 HERTZ TONE	42	THE	8A
20 MS SILENCE	43	TIME	8B
40 MS SILENCE	44	TRY	8C
80 MS SILENCE	45	UP	8D
160 MS SILENCE	46	VOLT	8E
320 MS SILENCE	47	WEIGHT	8F

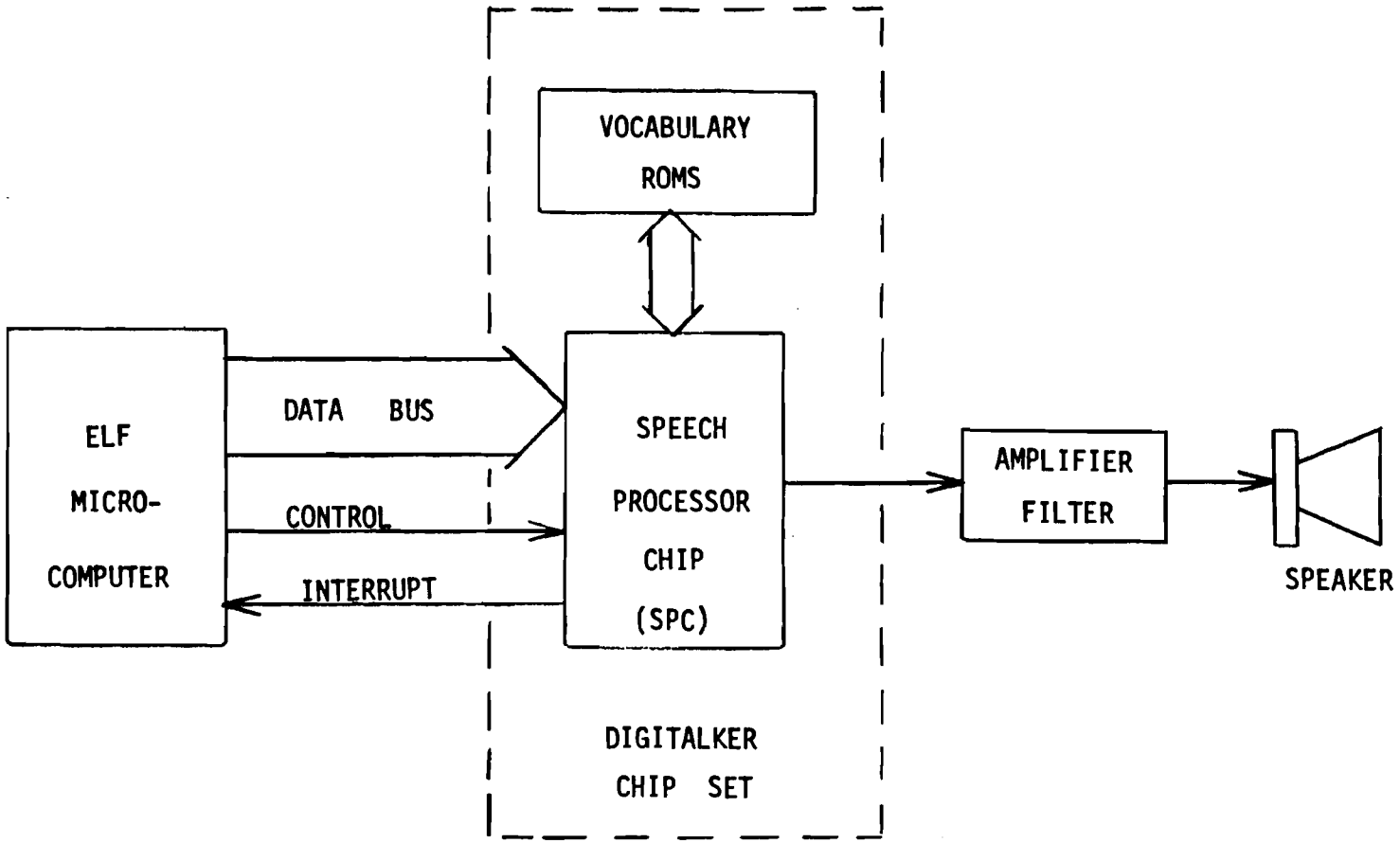


FIGURE 1

ELF/DIGITALKER BLOCK DIAGRAM

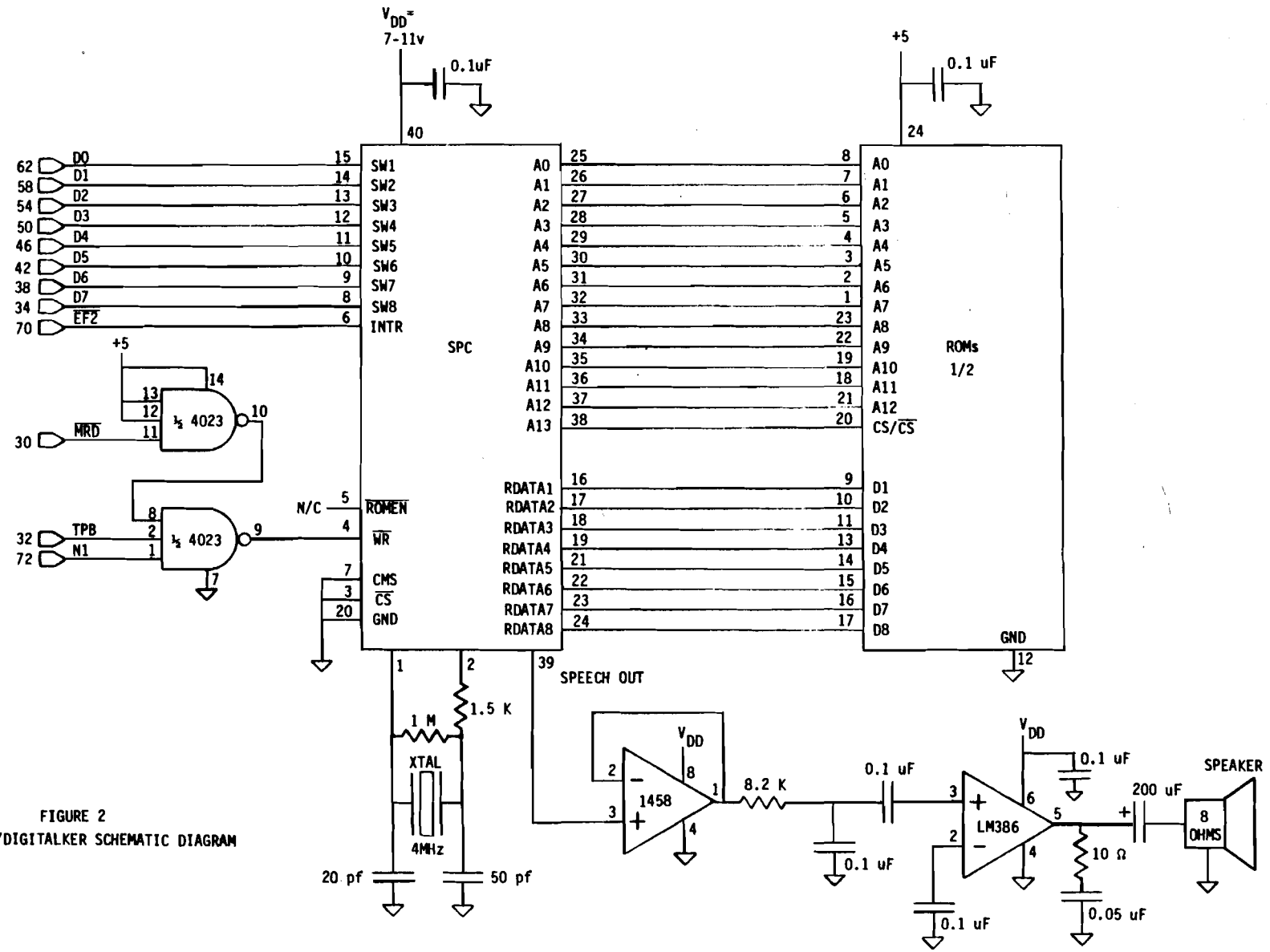
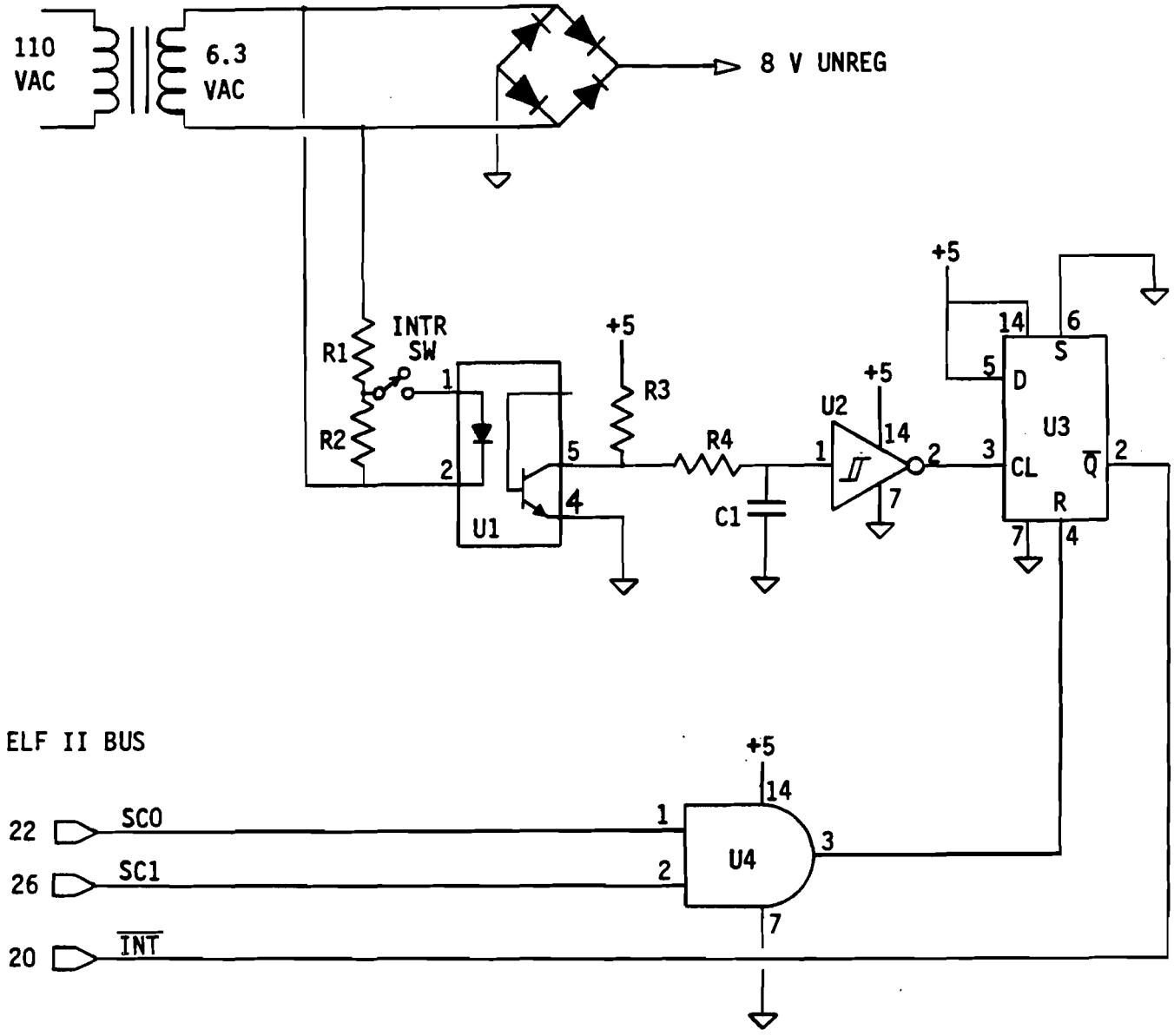


FIGURE 2
ELF/DIGITALKER SCHEMATIC DIAGRAM



U1	4N25
U2	74C914
U3	4013
U4	4081
R1	820 Ohms
R2	180 Ohms
R3	10 K
R4	22 K
C1	0.05 uF

FIGURE 3

TALKING CLOCK INTERRUPT CIRCUIT

TALKING CLOCK PROGRAM

ADDR	CODE	LABEL	MNEMONICS	COMMENTS
1E	F8 E6 A1		LDI INT PLO 1	INTRPT PROG CNTR
21	F8 1F A2		LDI 1F PLO 2	STACK POINTER
24	F8 ØF A7		LDI ØF PLO 7	MINUTE FLAG
27	A8		PLO 8	INTRPT COUNTER LO
28	F8 ØE B8		LDI ØE PHI 8	INTRPT COUNTER HI
2B	F8 D5 A9		LDI INC PLO 9	INC SUBR PNTR
2E	7A (OR 7B)		REQ (OR SEQ)	USE 7A BFOR 1PM
2F	F8 98 AA	INIT	LDI HOUR PLO A	HOUR POINTER
32	F8 9C AB		LDI MIN PLO B	MINUTE POINTER
35	F8 93 AC		LDI WORD PLO C	WORD POINTER
38	F8 A1 AD		LDI A/P PLO D	A/P POINTER
3B	EA		SEX A	X=A
3C	64 2A		OUT 4 DEC A	DISPLAY HOUR
3E	EB		SEX B	MINUTE POINTER
3F	87		GLO 7	FLAG CHECK
4Ø	3A 8E		BNZ ICHK	IF NZ BR ICHK
42	D9		SEP 9	INCREMENT MINUTE
43	FF 6Ø		SMI 6Ø	6Ø MINUTE CHECK
45	3A A6		BNZ ADJ	IF MINUTE = 6Ø
47	73		STXD	BCD MIN = Ø
48	F8 43 5B		LDI 43 STR B	TENS CODE = SILNCE
4B	6Ø 6Ø 73		IRX IRX STXD	MIN CODE = SILENCE
4E	EA D9		SEX A SEP 9	INCREMENT HOUR
5Ø	FF 12		SMI 12	IF HOUR NOT 12
52	3A 59		BNZ TEST	BRANCH TEST
54	ED FØ		SEX D LDX	GET A/P
56	FB ØF 73		XRI ØF STXD	REVERSE IT
59	EA	TEST	SEX A	X = HOUR POINTER
5A	FØ FF 13		LDX SMI 13	IF HOUR <> 13
5D	3A 6B		BNZ HCHK	BR HCHK
5F	31 64		BQ REQ	OTHERWISE
61	7B 3Ø 65		SEQ BR SET	REVERSE
64	7A	REQ	REQ	Q
65	F8 Ø1	SET	LDI Ø1	SET HOUR TO Ø1
67	5A 6Ø		STR A IRX	STORE BCD HOUR
69	3Ø 73		BR STOR	BRANCH STORE
6B	FØ FD Ø9	HCHK	LDX SDI Ø9	D = Ø9-D
6E	72 33 73		LDXA BPZ STOR	IF HOUR > 9
71	FF Ø6		SMI Ø6	REVERSE DEC ADJ
73	73	STOR	STXD	STORE TALK CODE
74	FØ 64		LDX OUT 4	SHOW BCD HOUR
76	31 7E		BQ NITE	CHECK HOUR OK
78	FF Ø7		SMI ON	IF TIME < ON
7A	3B 2F		BM INIT	BRANCH INIT
7C	3Ø 82		BR TALK	OTHWS BRANCH TALK
7E	FD 11	NITE	SDI OFF	IF T > OFF
8Ø	3B 2F		BM INIT	BRANCH INIT
82	EC FØ	TALK	SEX C LDX	GET WORD CODE
84	FD FF		SDI FF	IF CODE = FF
86	32 2F		BZ INIT	BRANCH INIT
88	62 6Ø		OUT 2 IRX	SPEAK WORD, RX+2
8A	35 8A	WAIT	B2 WAIT	WAIT FOR WORD
8C	3Ø 82		BR TALK	NEXT WORD

TALKING CLOCK PROGRAM (CONTINUED)

```
8E    3F 2F        ICHK    BN4 INIT          I PRESS?
90    64                   OUT 4             SHOW MINUTE
91    30 82                BR TALK           BRANCH TALK
93    8A 8A        WORD    THE               ANNOUNCE
95    8B 8B                TIME              TIME
97    60                   IS                PHRASE
98    11           HOUR    11                BCD HOUR
99    0B 0B                0B                HOUR TALK CODE
9B    17                   FIFTY             TENS TALK CODE
9C    55           MIN     55                BCD MINUTE
9D    05 05                FIVE              MINUTE TALK CODE
9F    46 46                SILENCE           PAUSE
A1    20 20        A/P     A                 A=20, P=2F
A3    2C 2C                M                 M
A5    FF                   FF                END PHRASE CODE
A6    F0 73        ADJ     LDX STXD          GET BCD MINUTE
A8    FF 20                SMI 20            IF MINUTE < 20
AA    3B B5                BM SILN           BRANCH SILN
AC    F6 F6                SHR SHR           OTHERWISE SHIFT
AE    F6 F6                SHR SHR           MSD TO LSD
B0    FC 14 5B             ADI 14 STR B      FORM TENS & STORE
B3    30 C8                BR UNIT           BRANCH UNITS
B5    F8 43 5B     SILN    LDI 43 STR B      TENS = SILENCE
B8    60 F0                IRX LDX           GET BCD MINUTE
BA    FD 09                SDI 09            IF MINUTE < 10
BC    33 C4                BPZ TENS          BRANCH TENS
BE    72                   LDXA              OTHWS TALK CODE =
BF    FF 06 73             SMI 06 STXD       REV DEC ADJ (HEX)
C2    30 2F                BR INIT           BRANCH INIT
C4    2B           TENS    DEC B             POINT TO TENS
C5    F8 2E 5B             LDI 2E STR B      TENS = OWE
C8    60 72        UNIT    IRX LDXA          GET BCD MIN
CA    FA 0F 5B             ANI 0F STR B      STRIP TENS
CD    3A 2F                BNZ INIT          IF BCD UNITS =0
CF    F8 43 73             LDI 43 STXD       UNITS TALK CODE =
D2    30 2F                BR INIT           SILENCE
D4    D0           EXT1    SEP 0             MAIN POINTER
D5    F0 FC 01     INC     LDX ADI 01        INCREMENT TIME
D8    A7                   PLO 7             SET MINUTE FLAG
D9    FA 0F                ANI 0F            STRIP TENS
DB    FD 09                SDI 09            DECIMAL ADJ RQD?
DD    87                   GLO 7             IF YES DF = 0
DE    CF                   LSDF              SKIP IF NO DEC ADJ
DF    FC 06                ADI 06            DECIMAL ADJUST
E1    73 60                STXD IRX          STORE TIME
E3    30 D4                BR EXT1           RETURN
E5    70           EXT2    RET               RESTORE X,P; R2+1
E6    22           INT     DEC 2             DECREMENT STACK
E7    78                   SAVE              SAVE OLD X,P
E8    22 52                DEC 2 STR 2       SAVE D
EA    88 3A FB             GLO 8 BNZ DCR     BR IF R8.0 NOT ZRO
ED    98 3A FB             GHI 8 BNZ DCR     BR IF R8.1 NOT ZRO
F0    F8 0F A8             LDI 0F PLO 8      LOAD R8
F3    F8 0E B8             LDI 0E PHI 8      =0E0F
F6    F8 00 A7             LDI 00 PLO 7      RESET MINUTE FLAG
F9    30 FC                BR RSTO           BRANCH RESTORE
FB    28           DCR     DEC 8             DECREMENT R8
FC    72 30 E5     RSTO    LDXA BR EXT2      RESTORE D AND EXIT
```

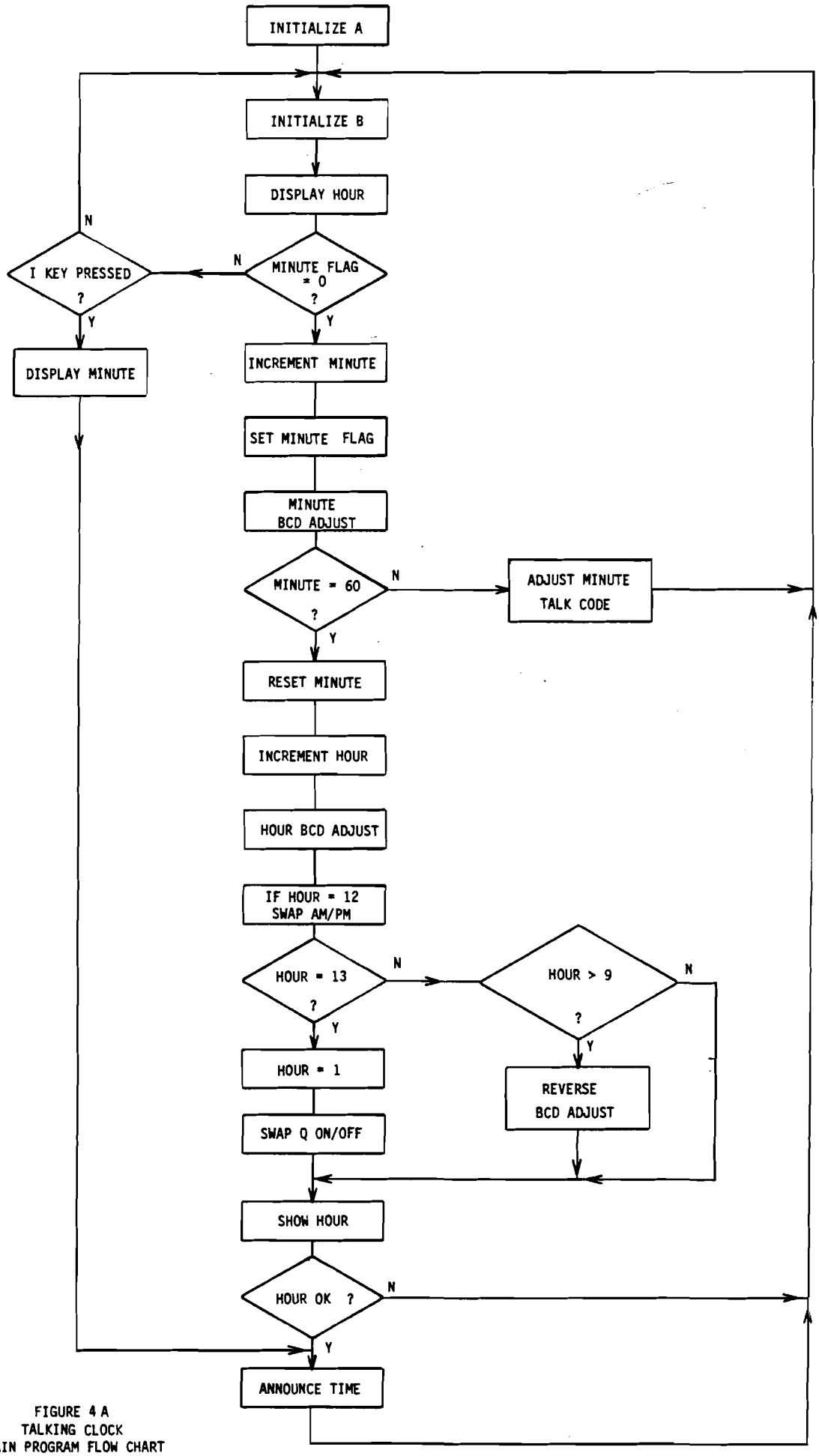


FIGURE 4 A
TALKING CLOCK
MAIN PROGRAM FLOW CHART

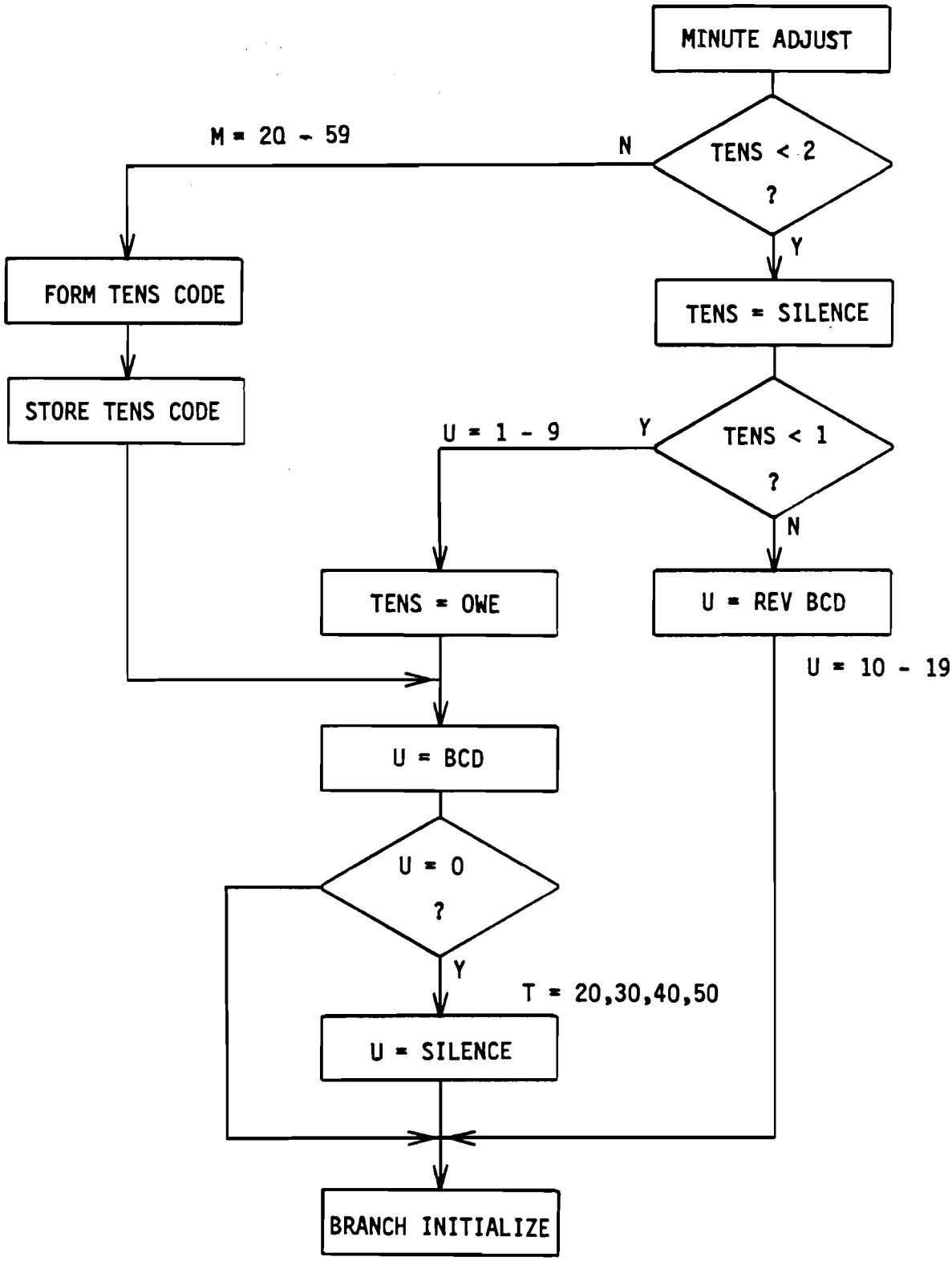


FIGURE 4B
TALKING CLOCK
MINUTE TALK CODE ADJUST FLOW CHART

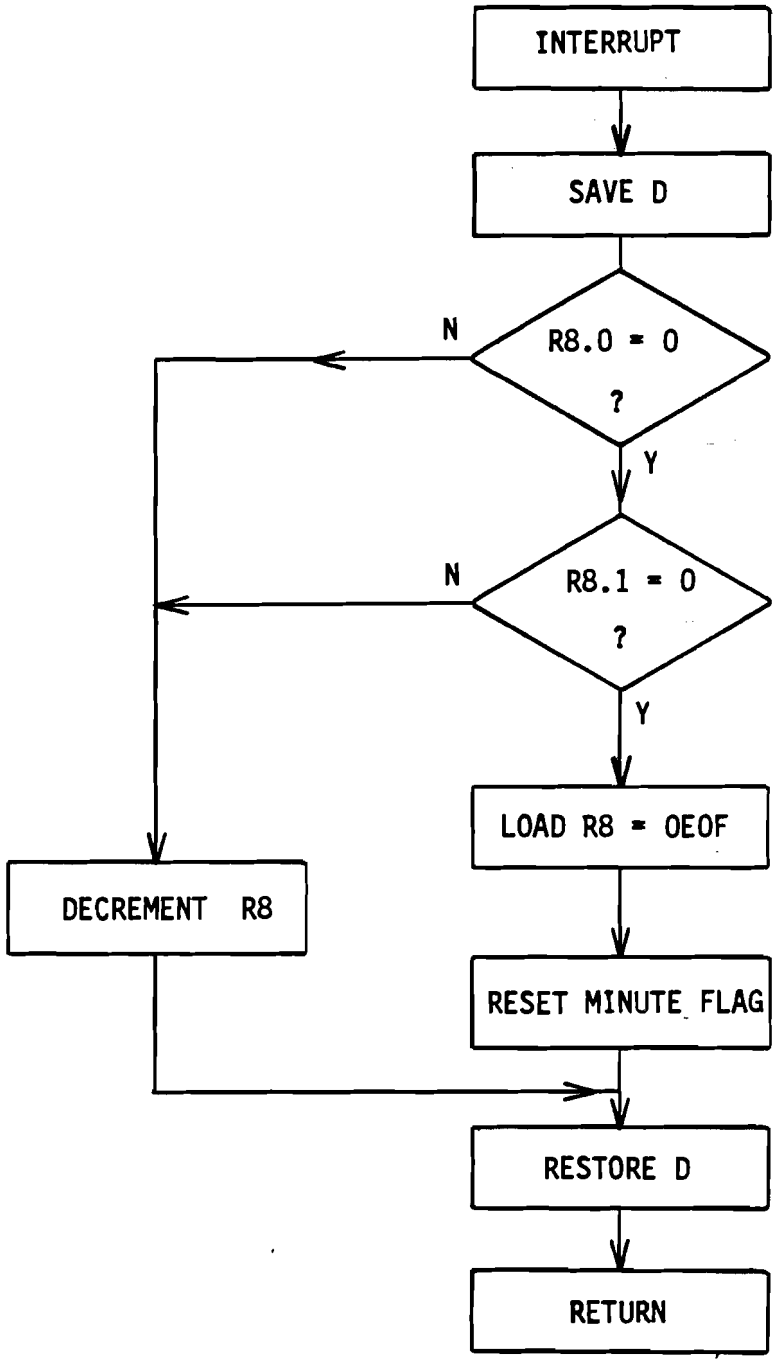


FIGURE 4C
TALKING CLOCK
INTERRUPT FLOW CHART

PROGRAM 2

DIGITALKER TIMER

```
ADDR  CODE       LABEL   MNEMONICS          COMMENTS

20    F8 80 A4   INIT    LDI 80, PLO 4      SET POINTERS
23    F8 90 A5           LDI 90, PLO 5      TO PHRASES
26    F8 A0 A6           LDI A0, PLO 6
29    F8 B0 A7           LDI B0, PLO 7
2C    F8 C0 A8           LDI C0, PLO 8
2F    F8 D0 A9           LDI D0, PLO 9
32    F8 E0 AA           LDI E0, PLO A
35    F8 F0 AB           LDI F0, PLO B
38    F8 03 AC           LDI 03, PLO C      TEST VALUE
3B    F8 6D AD           LDI SHOW, PLO D    RD POINTS TO SHOW
3E    ED                 SEX D              X=D
3F    1C         RNDM    INC C              PUT RNDM NO. IN RC
40    3F 3F              BN4 RNDM           (CHOOSE PHRASE)
42    8C                 GLO C              GET RC
43    FA 07              ANI 07             RESTRICT RANGE
45    F9 E0              ORI E0             TO E4---EB
47    FC 04              ADI 04
49    73                 STXD               SHOW IT
4A    60                 IRX
4B    64                 OUT 4
4C    50                 STO 0              EXECUTE IT
4D    50                 STO 0              FILLER CMD
4E    F8 XX AC           LDI XX, PLO C      SET DELAY PER
51    F8 YY BC           LDI YY, PHI C      FOLLOWING TABLE
54    F8 ZZ AD           LDI ZZ, PLO D
57    2C         LOOP    DCR C              DELAY  XX YY ZZ
58    8C                 GLO C              3 MIN  5D 6A 67
59    3A 57              BNZ LOOP           5 MIN  9C B0 AB
5B    2C                 DCR C              6 MIN  BC D3 CD
5C    9C                 GHI C              MIN    01 01 01
5D    3A 57              BNZ LOOP           MAX    00 00 00
5F    2D                 DCR D
60    8D                 GLO D
61    3A 57              BNZ LOOP           END DELAY
63    F0         GTWD    LDX                GET WORD CODE
64    FD FF              SDI FF             STOP IF CODE = FF
66    32 20              BZ INIT
68    62                 OUT 2              OUTPUT WORD, RX+1
69    35 69      WAIT    B2 WAIT            WAIT TILL DONE
6B    30 63              BR GTWD            NEXT WORD
6D               SHOW                       SCRATCHPAD MEMORY

80    8A 03 6E 81 45 60 75 FF          E4   FIRST PHRASE CODE
90    8A 8B 60 8D FF                   E5   SECOND PHRASE
A0    61 60 8B 02 88 78 FF             E6
B0    78 88 2D 2E 36 FF                E7
C0    34 31 3D 8A 8B 66 FF             E8
D0    00 47 28 22 34 31 7F 02 88 FF    E9
E0    61 60 8B 02 49 8A 70 34 5A 80 FF EA
F0    01 02 03 88 FF                   EB   LAST PHRASE
```

REVIEW RCA LSI PRODUCTS - APPLICATIONS

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

RCA LSI Products - Applications SSD-280
RCA staff
RCA Solid State/Technical Publications
Somerville, NJ: 1982
336 pages, softcover
$7.00

While titled an LSI book it is almost entirely devoted to microprocessors and their peripherals. This book contains nearly fifty application notes ranging in complexity from interfacing CMOS to other logic families and an introduction to microprocessors, to color video and microcomputer controlled production line testing. Although hard engineering facts are the subjects of most of the notes, when to choose microprocessors over small-scale integration and programmable logic arrays, hardware/software trade-offs, and when and how to incorporate assembly language routines into PL/M are also discussed. Emphasizing its own microprocessor products family, RCA strays far enough to discuss programming the 2732 EPROM and interfacing the Intel 8253 programmable counter/timer. Other contents include CMOS ROM's and RAM's, reset and clock circuits, parallel interfacing, analog to digital conversion, keyboard interfacing, UART and serial interfacing, realtime clocks, uP's in color TV receivers, sixteen bit operations, uP controlled thermostats, cassette program storage, battery powered RAM's, hardware multiply and divide, 1804 and 1805 uP's, and CRT terminals.

While aimed at the engineer, this book will easily be understood by the experienced hobbyist and could serve as inspiration for new projects. As a vocational level manual for interfacing lab this book emphasizes real applications.

CHIP 8 AE Disassembler

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

The CHIP 8 program writen by RCA and the updated version developed by Larry Owen, Tony Hill and myself, offer a variety of game and programming opportunities for the 1802.

But, the code is bothersome to learn, being two butes long, and containing a variable or an address. The following program automatically sorts out the variables and addresses indicates the condition of the command, and prints out a statement on a 32 character line. The program is located at 2000h, used SCRT and several routines out of my monitor, SYMON, the first two pages of which is also listed for your reference if you so not use it.

The program is entered at 2000h with the PC = R3, SCRT and the stack set. The program sets the start of the program at 0200h, and asks for the end. Input the end in HEX, and sit back while it prints out the code. Please note that this version does not format the output to a printer (ie form feed), but rather outputs a continuous series of lines until done.

The disassembler decodes all origional CHIP 8 commands, but only the most useful (in my opinion) of the new CHIP 8 AE commands. For the adventuresome however, I have added a default long jump in the commands with non- decoded commands, so that more can be added as desired. These occure in 0xxx, 5xyn, 8xyn,9xyn, exnn, and fxnn. Just change the long jump to point to the routine for the new commands, as required.

A sample of the disassemblers output is listed below, listing all commands that it can decode. For your convenience, a listing of the CHIP 8 AE commands is shown on p.40.

```
0200  007F  RESET GAME
0202  00E0  CLEAR SCREEN
0204  00EE  ILS RETURN
0206  0123  CALL MLS AT 0123
0208  1234  GOTO 0234
020A  2345  CALL ILS AT 0345
020C  34AA  SKIP IF V4 EQ AA
020E  45CC  SKIP IF V5 NEQ CC
0210  6789  SET V7 EQ 89
0212  7801  SET V8 EQ V8 + 01
0214  8910  SET V9 EQ V1
0216  8021  SET V0 EQ V0 OR V2
0218  8232  SET V2 EQ V2 AND V3
021A  8453  SET V4 EQ V4 XOR V5
021C  8564  SET V5 EQ V5 + V6
021E  8675  SET V6 EQ V6 - V7
0220  8786  SET V7 EQ V7 / 2
0222  8897  SET V8 EQ V9 - V8
0224  8A2E  SET VA EQ V2 * 2
0226  9010  SKIP IF V0 NEQ V1
0228  AC09  POINT I AT 0C09
022A  B389  GOTO 0389 + V0
022C  C43F  SET V4 RND/3F
022E  D238  SHOW 8 AT V2,V3
0230  E29E  SKIP IF V2 EQ KEYBD
0232  E4A1  SKIP IF V4 NEQ KEYBD
0234  F007  SET V0 EQ TIMER
0236  F10A  SET V1 EQ HEX
0238  F40E  SET V4 EQ ASCII
023A  F415  SET TIMER EQ V4
023C  F518  SET BEEPER EQ V5
023E  F21E  SET I EQ I + V2
0240  FC29  POINT I TO LSD OF VC
0242  FA2C  POINT I TO MSD OF VA
0244  F333  SET I EQ DEC V3
0246  F155  SAVE V0 TO V1 AT I
0248  F065  GET V0 TO V0 FROM I
024A  FC95  POINT I TO VC FOR ASCII
024C  0000  ?
024E  0000  ?
```

CHIP 8 AE Disassembler

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

```
2000  F8 02 B8 F8 00 A8 D4 C1 1F D4 C1 44 D4 C1 54 00
2010  9B AB D4 C1 54 02 8B FA F0 D4 20 57 20 27 2A D4
2020  C1 65 3A 09 3E 24 30 00 20 71 10 20 C4 20 20 D6
2030  30 20 E7 40 21 0E 50 21 1D 60 21 34 70 21 45 80
2040  21 5B 90 21 FC A0 22 11 B0 22 22 C0 22 2E D0 22
2050  3C E0 22 5B F0 22 87 46 B9 46 A9 86 73 96 73 E9
2060  9F F3 32 69 19 19 19 30 60 E2 19 49 B6 09 A6 9F
2070  D5 8B 32 85 D4 C1 87 43 41 4C 4C 20 4D 4C 53 20
2080  41 54 A0 30 CC 9B FB 7F 3A 99 D4 C1 87 52 45 53
2090  45 54 20 47 41 4D 45 A0 D5 9B FB E0 3A AF D4 C1
20A0  87 43 4C 45 41 52 20 53 43 52 45 45 4E A0 D5 9B
20B0  FB EE CA 23 E0 D4 C1 87 49 4C 53 20 52 45 54 55
20C0  52 4E A0 D5 D4 C1 87 47 4F 54 4F A0 8B FA 0F D4
20D0  FE 15 9B C0 FE 15 D4 C1 87 43 41 4C 4C 20 49 4C
20E0  53 20 41 54 A0 30 CC D4 20 F3 D4 C1 87 20 45 51
20F0  A0 30 D2 D4 C1 87 53 4B 49 50 20 49 46 20 D6 C4

2100  8B FA 0F FF 0A C7 FC 07 FC 3A BF C0 FE 06 D4 20
2110  F3 D4 C1 87 20 4E 45 51 A0 9B C0 FE 15 9B FA 0F
2120  CA 23 E0 D4 20 F3 D4 C1 87 20 45 51 20 D6 9B F6
2130  F6 F6 F6 30 01 D4 21 3B C0 20 EA D4 C1 87 53 45
2140  54 20 D6 30 00 D4 21 3B D4 C1 87 20 45 51 20 D6
2150  D4 21 00 D4 C1 87 20 2B A0 30 19 D4 21 3B D4 C1
2160  87 20 45 51 20 D6 9B FA 0F AF FB 00 3A 70 30 2E
2170  8F FB 01 3A 82 D4 21 00 D4 C1 87 20 4F 52 20 D6
2180  30 2E 8F FB 02 3A 95 D4 21 00 D4 C1 87 20 41 4E
2190  44 20 D6 30 2E 8F FB 03 3A A8 D4 21 00 D4 C1 87
21A0  20 58 4F 52 20 D6 30 2E 8F FB 04 3A B9 D4 21 00
21B0  D4 C1 87 20 2B 20 D6 30 2E 8F FB 05 3A CA D4 21
21C0  00 D4 C1 87 20 2D 20 D6 30 2E 8F FB 06 3A DA D4
21D0  21 00 D4 C1 87 20 2F 20 B2 D5 8F FB 07 3A EB D4
21E0  21 2E D4 C1 87 20 2D 20 D6 30 00 8F FB 0E CA 23
21F0  E0 D4 21 2E D4 C1 87 20 2A 20 B2 D5 9B FA 0F CA

2200  23 D0 D4 20 F3 D4 C1 87 20 4E 45 51 20 D6 C0 21
2210  2E D4 C1 87 50 4F 49 4E 54 20 49 20 41 54 A0 C0
2220  20 CC D4 20 C4 D4 C1 87 20 2B 20 56 B0 D5 D4 21
2230  3B D4 C1 87 20 52 4E 44 AF C0 21 19 D4 C1 87 53
2240  48 4F 57 A0 9B D4 21 01 D4 C1 87 20 41 54 20 D6
2250  D4 21 00 D4 C1 87 2C D6 C0 21 2E 9B FB 9E 3A 70
2260  D4 20 F3 D4 C1 87 20 45 51 20 4B 45 59 42 C4 D5
2270  9B FB A1 CA 23 E0 D4 20 F3 D4 C1 87 20 4E 45 51
2280  20 4B 45 59 42 C4 D5 9B FB 07 3A 9C D4 21 3B D4
2290  C1 87 20 45 51 20 54 49 4D 45 D2 D5 9B FB 0A 3A
22A0  AF D4 21 3B D4 C1 87 20 45 51 20 48 45 D8 D5 9B
22B0  FB 0E 3A C4 D4 21 3B D4 C1 87 20 45 51 20 41 53
22C0  43 49 C9 D5 9B FB 15 3A DD D4 C1 87 53 45 54 20
22D0  54 49 4D 45 52 20 45 51 20 D6 C0 21 00 9B FB 18
22E0  3A F7 D4 C1 87 53 45 54 20 42 45 45 50 45 52 20
22F0  45 51 20 D6 C0 21 00 9B FB 1E CA 23 11 D4 C1 87
```

```
2300  53 45 54 20 49 20 45 51 20 49 20 2B 20 D6 C0 21
2310  00 9B FB 29 3A 2F D4 C1 87 50 4F 49 4E 54 20 49
2320  20 54 4F 20 4C 53 44 20 4F 46 20 D6 C0 21 00 9B
2330  FB 2C 3A 4D D4 C1 87 50 4F 49 4E 54 20 49 20 54
2340  4F 20 4D 53 44 20 4F 46 20 D6 C0 21 00 9B FB 33
2350  3A 66 D4 C1 87 53 45 54 20 49 20 45 51 20 44 45
2360  43 20 D6 C0 21 00 9B FB 55 3A 86 D4 C1 87 53 41
2370  56 45 20 56 30 20 54 4F 20 D6 D4 21 00 D4 C1 87
2380  20 41 54 20 C9 D5 9B FB 65 3A A7 D4 C1 87 47 45
2390  54 20 56 30 20 54 4F 20 D6 D4 21 00 D4 C1 87 20
23A0  46 52 4F 4D 20 C9 D5 9B FB 95 CA 23 E0 D4 C1 87
23B0  50 4F 49 4E 54 20 49 20 54 4F 20 D6 D4 21 00 D4
23C0  C1 87 20 46 4F 52 20 41 53 FF 49 C9 D5 00 00 00
23D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
23E0  D4 C1 87 BF D5 00 00 00 F8 20 B3 F8 00 A3 D3 00
23F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

SYMON Initialization and Routine pages for CHIP 8 AE DISASSEMBLER

The following code is called in part by various routines.

```
C000  C0 C0 18 D4 C7 F0 C0 C6 00 C0 C1 D7 C0 C1 D1 C0
C010  C0 43 C0 C0 BD C0 C0 E8 F8 C0 B3 F8 1F A3 D3 E3
C020  71 23 F8 FE B2 B7 F8 FF A2 BC 73 93 B4 B5 F8 AC
  30  A4 F8 9D A5 D4 C6 3B D4 C1 C1 18 D4 C6 5A D4 C3
C040  31 30 4A D4 FE 0C D4 C1 87 BF D4 FE 0C D4 C1 87
C050  4D 3E 87 D4 FE 03 D4 C1 8F C0 5D 30 3E 3F C2 59
C060  42 D0 00 43 C2 34 44 C3 80 45 C3 00 46 C2 25 47
C070  DC 00 4A C2 0E 4D C2 16 50 C2 A9 51 C6 94 52 C2
C080  05 54 C1 DE 56 C6 44 57 C2 CD 58 C6 30 59 C2 84
C090  5A C2 00 00 FE 0F C0 C0 C0 C0 C0 C0 D3 BF E2 96
C0A0  B3 86 A3 12 42 B6 02 A6 9F 30 9C D3 BF E2 86 73
C0B0  96 73 83 A6 93 B6 46 B3 46 A3 9F 30 AB D4 FE 03
C0C0  FF 41 3B DC FF 06 33 E4 FE FE FE FE FC 08 FE AF
C0D0  8D 7E AD 9D 7E BD 8F FE 3A CF 30 E6 FC 07 33 E4
C0E0  FC 0A 33 C8 FC 00 9F D5 AF F6 F6 F6 F6 D4 C0 F4
C0F0  BD 8F FA 0F FC F6 C7 FC 07 FF C6 AD D4 FE 06 D5

C100  D4 FE 0C D4 C1 87 53 54 41 52 54 20 41 44 44 52
C110  45 53 53 20 3D A0 30 B9 D4 C1 00 9D B8 8D A8 D4
C120  FE 0C D4 C1 87 45 4E 44 20 BD D4 C1 59 0B D4 C1
C130  B9 8D 22 52 88 F5 AA 9D 52 98 75 BA 52 8A F4 12
C140  C2 FE 0F D5 D4 FE 0C 98 D4 FE 15 88 D4 FE 15 D4
C150  C1 59 02 D5 48 BB D4 FE 15 46 A9 89 32 53 D4 C1
C160  87 A0 29 30 5B 2A 9A 3A 6A 8A D5 87 73 F8 FF A7
C170  07 FB FF 32 83 2E 8E 3A 83 C8 87 73 D4 C1 87 8C
C180  F8 31 AE 60 F0 A7 D5 46 D4 FE 06 FE 3B 87 D5 46
C190  BB 46 AB 86 73 96 73 EB 9F F3 32 A2 1B 1B 1B 0B
C1A0  3A 98 E2 1B 4B B6 0B A6 9F D5 D4 FE 0C D4 C1 87
  B0  44 41 54 41 BF D4 C1 59 0D D4 FE 12 FB 0D 3A B9
C1C0  D5 96 B8 F8 00 A7 A8 46 AA 48 57 17 2A 8A 3A C9
C1D0  D5 D4 C1 87 0A 8D D5 FC 00 3E DD FF 00 D5 87 73
C1E0  F8 FF A7 07 3A E9 F8 FF C8 F8 00 57 60 F0 A7 D5
C1F0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00
```

CHIP 8 MINI AE INSTRUCTION SET

007F Reset program - PC = R0 = 1000

00E0 Erase the display

00EE Return from interpreter subroutine (2 mmm command)

0MMM Do a machine code subroutine at location 0MMM (subroutine must end with D4)

1MMM Go to 0MMM; (range 0200-0FFF)

2MMM Do an interpreter subroutine at location 0MMM (subroutine must end with 00EE)

3XKK Skip if VX = KK

4XKK Skip if VX = KK

5XY0 Skip if VX = VY

6XKK Set VX = KK

7XKK Set VX = VX + KK

8XY0 Set VX = VY

8XY1 Set VX = VX "or" VY (Note that VF is changed)

8XY2 Set VX = VX "and" VY (Note that VF is changed)

8XY3 Set VX = VX "xor" VY

8XY4 Set VX = VX + VY (Note that VF becomes 00 if the sum is less than or equal to FF and 01 if the sum is greater than FF)

8XY5 Set VX = VX - VY (Note that VF becomes 00 if VX is less than VY and 01 if VX is greater than or equal to VY)

8XY6 Set VX = VY divided by 2

8XY7 Set VX = VY - VX

8XYE Set VX = VY multiplied by 2

9XY0 Skip if VX = VY

AMMM Point I at 0MMM (range 0200 - 0FFF)

BMMM Go to 0MMM + V0

CXKK Set VX to a random byte; random byte is anded against KK first

DXYN Display N byte pattern at coordinates VX, VY; I (memory pointer) gives starting address of locations to be displayed. The displayed locations are exclusively ored against display field. VF becomes 01 if some of the display field is already set, 00 if it is not

EX9E Skip if VX = hex key input

EXA1 Skip if VX = hex key input

FX07 Set VX to the value of the timer; timer is counted down in interrupt routine

FX0A Set VX = keyboard (HEX)

FX0E Set VX = keyboard (ASC II)

FX15 Set timer to VX; timer is counted down interrupt routine so 01 is ca. 1/60 th second

FX18 Set tone duration to VX; turns Q on for duration specified by VX, 01 is ca. 1/60 th second

FX1E Set I = I + VX

FX29 Point I to pattern for least significant digit of VX

FX2C Point I to pattern for most significant digit of VX

FX33 Convert VX to decimal; 3 decimal digits are stored at M(I), M(I+1), and M(I+2), I does not change

FX55 Save V0 through VX in memory at locations specified by I, V0 at M(I), V1 at M(I+1), etc., I becomes I+X+1

FX65 Transfer memory locations specified by I to variables V0 through VX, V0 becomes M(I), V1 becomes M(I+1), etc., I becomes I+X+1

FX95 Point I to pattern of ASC II character stored in VX

NAME: _____ DATE: _____

PRODUCT ORDER	QUANTITY	UNIT PRICE	TOTAL
CPU Board	_____	$50.00	_____
Backplane and I/O Board, Ver. 2	_____	40.00	_____
Front Panel (with EPROM Burner, Clock)	_____	35.00	_____
VDU Board, Ver. 2	_____	40.00	_____
64K Dynamic (4116) Board	_____	50.00	_____
Netronics - Ace Adapter Board	_____	25.00	_____
I/O Adapter for Backplane, Ver. 1	_____	20.00	_____
Disk Controller Board, Ver. 2 (Available Aug. 84)	_____	50.00	_____
80 x 25 Video Board (Available Sept. 84)	_____	50.00	_____

Software - all cassettes supplied are in Netronics Cassette format

Fig FORTH - 6K - 0000H	_____	$10.00	_____
Tiny Pilot - 2K - 0000H	_____	10.00	_____
SYMON - 2K - 0000H	_____	10.00	_____
SYDOS - 4K - D000H	_____	10.00	_____
CHIP 8 AE - 1.5K - 1000H	_____	10.00	_____
FORTH - COSMIC Conquest 32K - 0000H	_____	10.00	_____

Back Issues

"Defacto" Year 1 - 3 (Edited)	_____	$20.00	_____
Year 4 Reprint	_____	10.00	_____
Year 5 Reprint	_____	10.00	_____
Year 6 Reprint	_____	10.00	_____
Year 7 Reprint	_____	10.00	_____

Membership - Year 7

Current Year - Sept. '84 - Aug. '85
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 promp 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
