Ipso Facto

ISSUE 25 OCT. 1981

INDEX PAGE

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

Executive Corner 2
Editors Corner 3
New Product Announcements (ACE Boards) 4
Members Corner 9

Of This and That 10
Roll Over Hex on the ELF II 11
Netronics Monitor Cassette Problems 11
Modifications to Ed McCormick's Monitor 11
CTL - The Fairly New Full Basic Command from Netronics 12
Simon ELF Update 12
Netronics Video Board Test Program 13
ELF II Cassette Hardware Modifications 16
Addendum - Mantei 2716 Eprom Programmer 16
Memory Test Program 17
Kingdom - A Tiny Basic Simulation Game 18
Quest Basic Printer Routine 23
Ratrace 1 - A Simulation Game in Quest Super Basic Ver. 3.0 . . 25
1802 Serial I/O Board 28
ACE Product Catalogue 45

Club Communique 59

1981/82 EXECUTIVE OF THE ASSOCIATION OF COMPUTER-CHIP EXPERIMENTERS

President:	John Norris	416-239-8567	Vice-President:	Ken Bevis	416-277-2495
Treasurer:	Mike Franklin	416-878-0740	Secretary:	Tony Hill	416-523-7368
Directors:	Bernie Murphy Fred Pluthero	416-845-1630 416-389-4070			
Newsletter:			Membership:	Bob Silcox Earle Laycock	416-681-2848
Production Manager:	Mike Franklin	416-878-0740	Program Convener:	Bernie Murphy Bert Dekat	
Editors:	Fred Feaver Steve Carter Bob Siddall Tony Hill		Tutorial/Seminars:	Ken Bevis Fred Feaver	
Advertizing:	Fred Pluthero	416-389-4070	Draughtsman:	John Myszkowski	
Publication:	Dennis Mildon John Hanson				
Hardware & R. and D.	Ken Bevis Don McKenzie Fred Pluthero Dave Belgrave	416-277-2495	Software:	Wayne Bowdish	416-388-7116
			Product Mailing:	Ed Leslie	416-528-3222

CLUB MAILING ADDRESS:

A.C.E.
c/o Bernie Murphy
102 McCraney Street East
Oakville, Ontario
Canada
L6H 1H6
Phone: 416-845-1630

CLUB MEETINGS:

Meetings are held on the second Tuesday of each Month, September through June at 7:30 in Room B123, Sheridan College, 1430 Trafalgar Road, Oakville, Ontario. A one hour tutorial proceeds each meeting. The college is located approximately 1.0 km north of the QEW, on the west side. All members and interested visitors are welcome.

ARTICLE SUBMISSIONS:

The majority of the content of Ipso Facto is voluntarily submitted by club members. While we assume no responsibility for errors nor for infringement upon copyright, the Editorial staff verify article content as much as possible. We can always use articles, both hardware and software, of any level or type relating directly to the 1802 or to micro computer components, periferals, products, etc. Please specify the equipment or support software upon which the article content applies. Articles which are typed are prefered, and usually printed first, while handwritten articles require some work. Please, please send original, not photocopy material. We will return photocopies of original material if requested. Photocopies usually will not reproduce clearly.

ADVERTISING POLICY

ACE will accept advertising for commercial products for publication in Ipso Facto at the rate of $25 per quarter page per issue with the advertiser submitting camera-ready copy. All advertisements must be pre-paid.

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 (ie Netronics Basic only, Quest Monitor required, requires 16K at 0000-3FFF etc.). The newsletter staff will attempt to publish Ipso Facto by the first week of: Issue 25 - Oct 81, 26 - Dec 81, 27 - Feb 82, 28 - Apr 82, 29 - Jun 82, and 30 - Aug 82. 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 6 issues of Ipso Facto published during the club year.

EDITOR'S CORNER

It is with pleasure, and a little trepidation, that I look forward to editing the fifth year's issues of IPSO FACTO. ACE is a major force in promoting the COSMAC 1802 micro computer, and it is with great satisfaction that I am able to contribute to that end. At the end of this issue is a "catalogue" of ACE products. We have very nearly created a complete advanced micro system. Only a new improved micro-board remains to be built. Our Disk Controller, Dynamic Memory and Video Display Unit Boards form the basis of an advanced micro system (and all were developed and produced by volunteer hobbiests, like you and me, not by the circuit engineering staff of the corporate giants!).

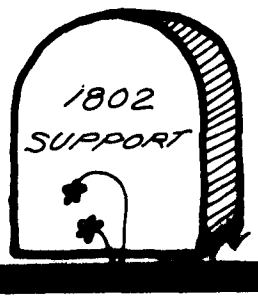
This year, the club proposes to develop the "serious" micro board with provision for upgrading to the 1804, 5 or 6 micros we understand will soon be available. In the next issue of I.F., an article will review the published data on the new clips to inform our members of their various capabilities. Also, the results of the membership questionaire will be reviewed. Preliminary observations show a marked increase in computer hardware, and in sophistication of software in use. It should make interesting reading.

Finally, it is sad to note the passing of yet another 1802 oriented business. Paul Piescik, owner of CUDDLY SOFTWARE was forced to suspend business due to poor financial/market performance. We regret the loss of a capable 1802 software source.

CUDDLY SOFTWARE 1981

BENCHMARK 1979

INFINITE 1978



HELP - FUTURE IPSO FACTO MAY BE 4 PAGES!

The Editor's file is now nearly empty of articles for future Ipso Facto issues. Hopefully, the 3 month Canadian postal strike delayed articles which would normally flow to the club. The editors do not write many articles in the newsletter--the content and the thickness of IPSO is up to you. We would be interested in publishing the following types of articles: application circuits, CLOCKS, thermometers, remote switches, sensors, A/D-D/A, modems, printer interfaces etc. software programs of interest--6847 (VDU) oriented programs for graphics or colour control, games, interfacing to Basic, Netronics Text Editor,--Quest Basic modifications for various printer devices, cassette, stringy or floppy interface improvements, Quest Basic application programs--games, routines or drivers for 1861 or machine language programs. But please--no more monitors!! I hope to be inundated with your articles soon!!!

NEW PRODUCT ANNOUNCEMENTS (ACE BOARDS) OCT 1, 1981

Two new boards for the ELF II user! ACE has now in stock a Quest Super Expansion Board to Netronics Ace Adapter Board "Interface" Board, and a DMA Adapter Board. The QNAA (Quest-Netronics-Ace Adapter) is designed to plug into one of the 86 pin slots of the NAB, and provide all appropriate signals to the Quest 50 pin interface. In addition, the Board offers switchable 4K memory decoding for the super expansion board Ram and 2 of the 3 Eprom Sockets (2716 Eproms) and a separate fixed 2K monitor enable plus boot and 1K Ram stack, addressed as F000-F7FF, and F800-FBFF respectively. The remaining 1K FC00-FFFF is left free for memory map purposes (VDU and disk). While it has not been proven, the QNAA should interface a Quest Super ELF Micro Board to the NAB to allow Quest users to use ACE boards too!

The DMA adapter board is a small 3 IC board designed to modify the ELF II DMA flip flop circuit (A17) to allow buss generated DMA demand signals. The Board was developed as a result of problems encountered with the ACE Disk Controller Board (which now works!!).

DYNAMIC MEMORY BOARD

Cheap memory is here, almost. Our TTL version 32 k Dynamic Board has been working relatively reliably for several months. However, relatively is not good enough!! The refresh circuitry, the heart beat of Dynamic memory, is susceptable to loading of the timing and MRD, MWR signals. (My board started acting up after I added 1 minor circuit). So, Don MacKenzie has built a CMOS version (fewer chips too!!) and the club is now ringing it out.

So far, it has proven very reliable. Don has modified the board to accept 64K of dynamic memory with a 4K memory shadow option. We expect delivery of orders in early december. Conditions/advance orders will be accepted now!!

In order to facilitate club members understanding of ACE's progress in developing an 1802 system, and to assist in ordering products, we are including a "catalogue" of our products at the end of this issue. We will issue updates as new products become available.

1016NP

RCA News

RCA/Solid State Division
Route 202
Somerville, N.J. 08876
(201) 685-6423

TWO NEW CHIPS FROM RCA HIGH-SPEED 1-OF-8 DECODER

A high-speed, high-drive-current decoder for memory or I/O addressing has been announced by RCA. This device, the CDP1873C, features active-low outputs for direct compatibility with active-low chip selects, and multiple enable inputs for expansion and additional control.

The CDP1873C can be used in CDP1800 or other general-purpose microprocessor systems as an address decoder or I/O address bus expander. The device is pin-compatible with the 74LS138, and can be used in the same types of applications, but where low power and high noise immunity are desirable.

Maximum propagation delay from any input to output line is 100 ns at 5V under the full commercial temperature range of -40 to +85°C, reducing the total access time path in a large decoded memory or I/O system. Output source and sink capability is balanced, and is equivalent to that required for four standard TTL loads. This high drive capability allows direct interfacing to large memory systems without additional buffering, and allows low propagation delays to be maintained even with heavy capacitive loads.

PROGRAMMABLE CMOS 8-LEVEL INTERRUPT CONTROLLER FEATURES VECTORED AND MASKED OPERATIONS

The new CDP1877 CMOS IC programmable interrupt controller from RCA Solid State Division is designed to minimize software and real-time overhead for multi-level priority interrupts in CDP1800-based microprocessor systems. The device features eight levels of prioritized interrupts and software-programmable vectoring to interrupt routines. The CMOS device also offers low-power dissipation of just 7.5 nw. The CDP1877 interrupt controller IC is a memory-mapped device with latched interrupt requests and hard-wired interrupt priorities.

Applications for CDP1877 include industrial controllers, traffic-control signal systems, burglar and fire alarm systems, medical electronics, engine controllers and systems in which real-time operation is required.

DATA BUS CONTENTION DURING CDP1802 REGISTER-TO-REGISTER OPERATIONS

In 1802 based systems using ROM's or EPROM's bus contention problems have been found to occur during internal data transfer operations (GHI, PHI, GLO, PLO). As a result, data is lost in one or more registers.

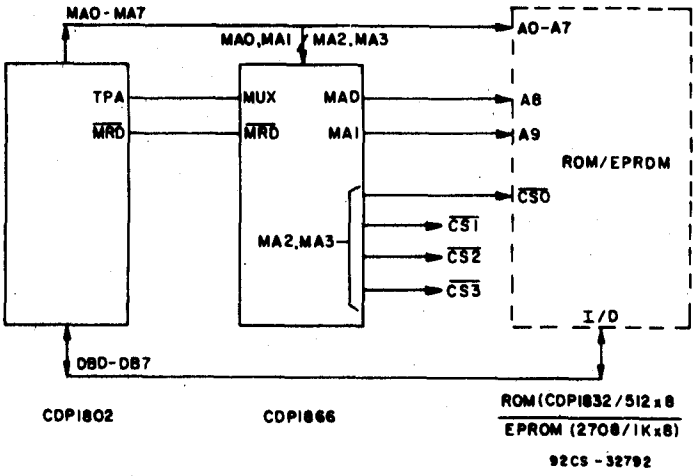
The 1802 generates a valid 16-bit address and a TPA signal during these operations. If the chip-select signals for the ROM or EPROM is only controlled by higher order address bits, then it is very probable that these memories can be selected and have their output drivers "turned on", creating a bus contention problem with the 1802 data bus drivers.

The solution to this problem is to either gate the chip-select functions with MRD externally, or find a spare input on the memories for MRD. During these register operations MRD is held high. (See table 1 on p. 90 of the MPM-201B Manual). See below for specific suggestions.

A. Wiring MRD to a Spare Input

- o CDP1834 (CS1 or CS2) - only if they are "active low"
- o 2716, 2732, 2758 (OE)

B. Gating MRD Externally with the Chip-Select Function



HELP - QUEST DYNAMIC MEMORY BOARD PROBLEMS

Several members have written indicating that they cannot get the Quest 64k board to work. Most difficulties seem to be encountered by those trying to add the board to the ELF II. Would any club member who has successfully modified the ELF II to get te board to work please write to ACE and let us know how you did it. Also, would all members who have had difficulty with the board please write detailing the problem (and solutions if found), so we can communicate the information to Quest.

NEW RCA MICROCOMPUTER DEVELOPMENT SYSTEM ONLY $499

Two new low-cost Microboard Computer Development Systems have been introduced by the RCA Solid State Division. At only $499 (single unit price), the CDP18S693 is the lowest cost 1802 Microprocessor Development System on the market. Complete with ROM-based floating point BASIC interpreter and system utility software, the system includes a CMOS single-board microcomputer, memory/audio cassette controller board, one audio cassette tape drive, a five-card chassis and case, and a 5-volt power supply.

The CDP18S694, priced at $799 (single unit price), includes all of the above plus a ROM-based 1802 assembler/editor PROM programer board, and a second cassette tape drive.

With the simple addition of a user supplied RS232 or 20 mA compatible terminal, the system allows the user to generate 1802 software for use in any microboard system. The system resident BASIC contains 1802 specific I/O statements, allowing the customer to write 100% of his applications program in the high level language. The addition of the 1802 assembler/editor allows the user the added flexibility of writing applications software in assembly level or mixing both BASIC and assembly level software as required.

FORTH INTEREST GROUP
P.O. Box 1105
San Carlos, CA 94070

FORTH INTEREST GROUP NATIONAL CONVENTION

The FORTH Interest Group (FIG) announced the scheduling of its National Convention to be held November 28, 1981, 9 am - 6 pm, at the Santa Clara Marriott Hotel. This one day convention will include presentations, workshops, hands-on equipment and a number of vendor exhibits. An evening dinner will follow the days activities and will feature a speaker. Registration is $3.00 with dinner extra.

The theme of the convention this year is education as related to the FORTH program applications. Papers are being solicited for presentation at the convention. Those wishing to present papers or exhibit should send a letter of intent to Convention Chairman, FORTH Interest Group, P.O. Box 1105, San Carlos, CA 94070. Authors' Instructions will be sent to individuals sending a letter of intent.

The FORTH Interest Group now has over 2500 members, worldwide (all 50 states and 36 foreign countries). Several chapters are meeting or in the process of forming. In keeping with the spirit of the FORTH language, several groups meet on the fourth Saturday of each month.

Mountain View Press
PO Box 4656
Mountain View, CA 94040

Mountain View Press, the FORTH source, has a wide variety of printed publications for the FORTH language. While no machine readable information is handled by Mountain View Press, there is an extensive list of materials available for the installation of FORTH. For $20.00, to cover the cost of an Installation Manual and a specific Source Listing, FORTH can be implemented on the purchaser's micro or mini computer.

A number of user and reference manuals and books about FORTH are also available from Mountain View Press. Some of these are: "Using FORTH", "A FORTH Primer", PDP-11 FORTH User's Guide", "Threaded Interpretive Languages" and "Tiny Pascal in fig-FORTH". New titles are added as they become available, either from authors or from other publishers.

The Photo/Electric Arts Foundation
People and the technologies of light/electricity

Box 7109, Postal Station A
Toronto, Ontario, M5W 1X8
Canada, (416) 367-0590

COMPUTER CULTURE '81, the only public event in the world to focus on the cultural aspects of computers and microelectronics, will be held in Toronto from Monday, November 16th to Sunday, November 22nd. The seven-day program, to take place at the Ontario College of Art, The Art Gallery of Ontario, the Flavio Belli Gallery, Gallery 76, the Music Gallery and the Village by the Grange, features displays and demonstrations, an open conference, an art show, a multi-media tribute to Marshall McLuhan, a day of computer animation and a computer music symposium. For further information contact: The Photo/Electric Arts Foundation, Box 7109, Postal Station A, Toronto, Ontario, M5W 1X8, or phone (416) 367-0590.

COMPUSERVE #70535,757
- by Jeff Wolf, 1219 Belden Road, Columbus, Ohio 43229

One of my periferals is a direct-connect modem. I am a member of the Compuserve information network. Perhaps Ipso-Facto could have a listing of ACE members who are a part of Compuserve, and list their I.D. numbers. Communication could become more efficient and software could be exchanged quite easily if a network such as Compuserve were utilized.

Interested members please contact me at the address listed below.

Jeff Wolf
Compuserve #70535,757
1219 Belden Road
Columbus, Ohio
43229

NETRONICS ELF II, two 4K boards, Giant board, ASCII keyboard, RF modulator, Expansion PS, Tiny Basic tape. $350.
Royal Dossett, 2795 Pheasant Rd, Excelsior, MN 55331
612-471-9252

1802 NEWSLETTERS.

Questdata (1/1-2/9, 21 issues)	$20
IPSO FACTO (1-23, 23 issues)	20
VIPER (1/1-2/10, 20 issues)	20
Club 1802 (5 and 8, 2 issues)	10
Netronics (1 and 2, 2 issues)	10
All of the above	40

Royal Dossett, 2795 Pheasant Rd, Excelsior, MN 55331
612-471-9252

BASIC ELF: 1K RAM, I/O is 2 Hex Displays, 8 toggle switches, and an 1861 video chip, built on Quest's PC board. 1.79 Mhz clock and 5v Pwr Supply. $70.00

4K of new 21L02 memory chips (32 pcs) $25.00

Also, have the following books:
"Pips for VIPs" by Swan (with cassette) $15.00
Pittman's "Short Course in Programming" (for Elf and Elf II) $4.00
Netronics "Tiny Basic Manual" with cassette $10.00
Quest "Tiny Basic" with object listing $4.00
Lancaster's "TV Typewriter Cookbook" $3.00
Intel's 8080 "Microcomputer System User's Manual" $2.00
Pollack & Sterling's "Guide to PL/1 $4.00
Barden's "How to Program Microcomputers" $2.00

Also have an Elf "Tiny Adventure" game written in Tiny Basic and requires 8K of user memory space. I need S-100 Static memory, a cassette recorder with tape counter, an acoustic modem, a working TTY or printer or???
Write Gary Jones, 7717 N. 46th Dr. Glendale Arizona 85301

FOR SALE - Netronics ELF II with Giant board, RF modilator, RCA manual CDP-1802, Pittmans Short Course in Programming, DEFACTO, and Ipso No. 13 -24.

$175.US., or best offer.

David G. Plank, RD 2, Box 193, New Milford, PA.' USA, 18834.

OF THIS AND THAT

- by M. Franklin

The following chart is useful when developing the logic flow of a circuit.

LOGIC INPUT	GATE LOGIC AND	NAND	OR	NOR	XOR	XNOR
0 0	L	H	L	H	L	H
0 1	L	H	H	L	H	L
1 0	L	H	H	L	H	L
1 1	H	L	H	L	L	H
CMOS	4081	4011	4071	4001	4030/70	4077
TTL 74	'08	'00	'32	'02	'86	'266

While most people know that 74C and 80C series IC's are CMOS and pin for pin compatible with 74 and 74 LS and 80 series chips, a large number of 4000 series chips are also pin for pin and functionally compatible with 74 series chips - ie:

7404	4069		hex inverter
74367	40097	8097	hex tri state buffer
74368	40098	8098	hex tri state inventory buffer
7486	4030/70		XOR
74266	4077		XNOR
74174/175	40174/5		HEX/QUAD D Flip Flop
74160/1/2/3	40160/1/2/3		BCD/Binary up counters
74192/3	40192/3		BCD up/down counter
74194/5	40194/5		4 bit shift register
74139	4555/6		Dual 1 of 4 decoders
74153	4539		Dual 4 in mux
74259	4099		8 bit addressable latch
7485	40085/4585		4 bit comparitor

ROLL OVER HEX ON THE ELF II

- by T. Jones, 409 Springdale Ave. Enterprise, Alabama, USA

The following changes to Netronics Monitor will generate the "roll over hex" feature requested by Bill Echel in Ipso Facto #16:

```
F029 6C 64 22 BYTEN - INP 4, OUT 4, DEC 2
F02C 3F 29          BN4 (BYTEN)
F02E 37 2E          B4 loop
```

NETRONICS MONITOR CASSETTE PROBLEMS

- by T. Jones -

I recently ran into intermittent tape read problems with my ELF II. By varying the value of FDBB between the original value "OD" and a maximum of "11" I vound a value of "OF" to give perfect performance. The change may compensate for mechanical differences between cassette players or tape quality.

MODIFICATIONS TO ED MCCORMICK'S MONITOR - IPSO FACTO #5, P 30

- by J. Stephens, 2324 Dennywood Dr., Nashville, Ten., USA 37214

I have made a few minor changes to the program which makes it load 4k instead of one page. The changes are as follows:

loc	change to
01	F8 10 B1
3F	F8 00 B1
98	FF 10
FA	FF 10

With these changes the Load or read function dumps 4k and takes about three minutes. I've found that you can push reset before the entire 4K is cycled thru and this does not affect the part that has been loaded.

CTL - THE FAIRLY NEW FULL BASIC COMMAND BY NETRONICS

- by A. Magnani, 22-68 28th St., Astoria, N.Y., USA 11105

Before I understood the CTL command in BASIC, I could not use the cursor control sequence in BASIC. Whenever I hit the sequence (ESC,=,?,?) (to position the cursor in the bottom left corner), it would echo back to the VID-1 and this made for a very messy program. I called Netronics, and they said to use the CTL with parenthesis around the decimal equivalent of the code for ESCape, L(clear screen), etc. I tried PRINT CTL(12)--to clear the screen, but this did not work. It instead output hqri. I tried PRINT CTL(27)--to output an ESC to initiate the cursor sequence, and it output hrwi. I then decided to find exactly what the CTL command is supposed to do; since Netronics was of no help. After some experimentation, I found that it takes the entire string following it, and adds a binary 1000000 to each character and then outputs the revised character. Depressing the CTL-key also accomplishes the same result. Therefore, to output an ASCII ESC, you can type in a PRINT CTL [, for an=, CTL , for a +, or CTL , etc. for any of the ASCII codes. For example, to move the cursor to 1,3 you would use "ESC","=","A","C", in direct mode, or, in indirectly mode, PRINT CTL[AC. I do not know if the version at Netronics works as they say with decimals, but mine sure doesn't. Using the above sequence I can now have the cursor control feature of the VID-1, without having my programs jump all around the screen upon listing them.

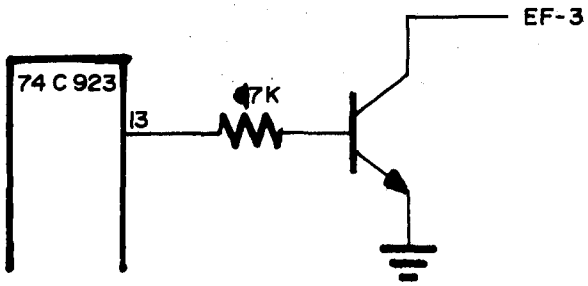
SIMON ELF UPDATE

- by J. Stewart, 6 Clearmount Circle, St. Catharines, Ont., L2T 2N9

After playing Richard Moffie's version of Simon Elf, I realized there had to be a better way than pushing the ELF II I button for every entry. After some experimentation, the following circuit modification to my ELF II keypad decoder worked. Change the Simon ELF code as follows:

003D 17 3E 3D 36 40

Note: If you use the Elf II monitor, you must also change the monitor routines as well.



NETRONIC'S VIDEO BOARD TEST PROGRAM

- by D. Bauer, RR#1, Denmark, Maine, USA 04022

When I got my Netronics ASCII terminal ready to run, I found I had no way to exercise it. (I didn't have the Tiny Basic tape yet.) A call to Netronics yielded a simple program to feed the keyboard through the computer to the video display board but no way to feed the display from computer generated data. I wrote the attached program to provide the required output.

The program can be run in three different modes. Enter (from the monitor or whatever) at 0003. Then input an 00, 01 or 02 command on the hex keypad. The 02 command calls up the Netronics program. The ASCII keyboard output is simply repeated by the computer to the video display. An 00 command calls up the HEX-KEY program. Now data input on the hex keypad is transmitted to the display. An 01 command calls up the AUTOMATIC program which will fill the display with all the printable characters.

'Automatic' takes about 1 minute, 17 seconds to write a full 16 line by 64 character display with a DS before each character. The DS cuts the effective transmission rate in half but it is needed to get the shifted characters. Since it doesn't interfere with the regular characters, it was easier to include it for all of them.

The most useful part of the program for those of us who must make do without a serial output in hardware is the XMIT subroutine. It takes the hex byte in the D register and transmits it in serial ASCII format via the Q line at 300 Baud. The timing constants are set up for my ELF II with its 3.58/2 Mhz. clock. The range given was found by trial and error on my system.

```
0000    C0    LBR
   1    F0    addr      jump to Monitor
   2    00    addr
   3    F8    LDI
   4    00    data
   5    B1    PHI 1
   6    B4    PHI 4
   7    F8    LKI
   8    29    addr      Set starting address of XMIT  Subroutine
   9    A1    PLO 1
   A    F8    LKI
   B    50    addr      Set X register address    (for HEX-KEY routine)
   C    A4    PLO 4
   D    E4    SEX 4     R(X) = R(4)
   E    3F    BN4       loop until I is pressed
   F    0E    addr
0010    6C    INP 4     read hex keyboard into MR(X) & D
   1    32    BX        if 00, branch to HEX-KEY   routine
   2    46    addr
   3    FF    SMI
   4    01    data
   5    32    BZ        branch to AUTOMATIC routine if input is 01
   6    51    addr
   7    3F    BN4
   8    17    addr
```

```
   9     7B     SEQ
   A     37     B4        otherwise, feed ASCII keyboard direct to
   B     1A     addr      video board.  No computer interaction.
   C     7A     REQ       (Courtesy Netronics R&D, Ltd.)
   D     30     BR
   E     17     addr
   F     C4     NOP
0020     7A     REQ
   1     F8     LDI
   2     02     data
   3     B3     PHI 3     delay at end of each byte transmitted
   4     23     DEC 3     gives approx. 6 msec between bytes
   5     93     GHI 3     (approx. 36 msc total cycle time)
   6     3A     BNZ
   7     24     addr
   8     D0     SEP 0     return to main program
   9     B2     PHI 2     store D in R(2.1)        Start XMIT Subroutine
   A     F8     LKI
   B     08     data      load bit counter
   C     A2     PLO 2
   D     7B     SEQ       Set Q for start/stop bit
   E     F8     LDI       load bit timer (works OK from 72 to 81)
   F     77     data
0030     A3     PLO 3     load bit timer (works OK from 72 to 81)
   1     23     DEC 3     decrement bit timer
   2     83     GLO 3
   3     3A     BNZ       loop until bit timer is zero
   4     31     addr
   5     82     GLO 2     R(2.0) is bit counter
   6     32     BZ        if zero, branch to 6 msec. delay
   7     20     addr         then back to main pgm.
   8     22     DEC 2     decrement bit counter
   9     82     GLO 2
   A     32     BZ        if zero, then end of byte.  branch back for
   B     2D     addr         stop bit.
   C     92     GHI 2     get input byte
   D     F6     SHR       shift LSB to DF
   E     B2     PHI 2     put shifted byte back in R(2.1)
   F     C7     LSN F     long skip if DF = 0
0040     7A     REQ       reset Q if DF = 1
   1     38     SKP
   2     7B     SEQ       set Q if DF = 0
   3     30     BR        branch back to bit timer
   4     2E     addr
   5     C4     NOP
   6     3F     BN 4      loop until I is pressed    Start HEX-Key
   7     46     addr
   8     6C     INP 4     read hex keypad into M R(X) & D, R(X) = R(4)
   9     64     OUT 4     out to hex display, R(X) + 1
   A     24     DEC 4     R(X)-1
   B     37     B4        loop until I is released (Note:  substitute C4's
   C     4B     addr         for continuous repeat while I is pressed)
   D     D1     SEP 1     call ASCII  XMIT subroutine
   E     30     BR
   F     46     addr
```

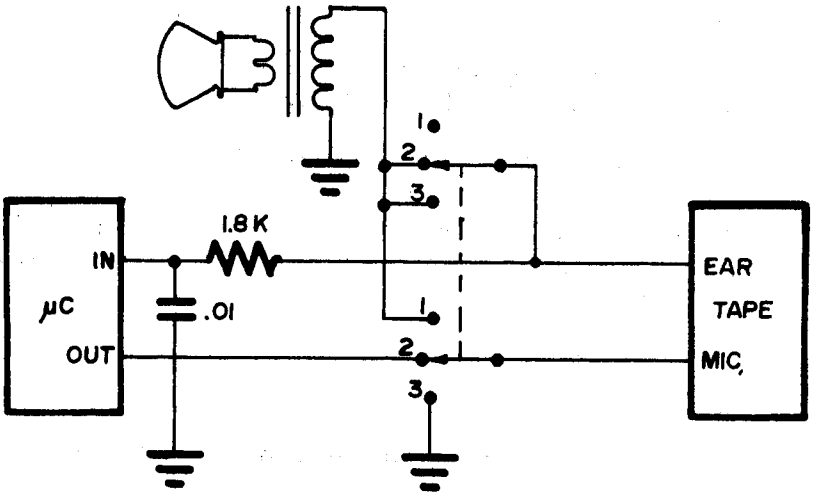
```
0050    X       X       X register work space
  1     F8      LDI                                     Start AUTOMATIC
  2     04      data    04 = home cursor
  3     D1      SEP 1   call XMIT
  4     F8      LDI
  5     00      data    set data generator to 00
  6     A5      PLO 5
  7     F8      LDI     set line position counter for 64 characters
  8     40      data    (for 32 character line, change to 20 HEX)
  9     A6      PLO 6
  A     F8      LKI
  B     10      data    01 = downshift
  C     D1      SEP 1   call XMIT
  D     85      GLO 5   get data to be sent
  E     D1      SEP 1   call XMIT
  F     15      INC 5   increment data generator
0060    26      DEC 6   decrement line position counter
  1     86      GLO 6
  2     3A      BNZ     if not end of line, branch back for next
  3     5A      addr    DS and character
  4     F8      LKI     if end of line, transmit carriage return and line
  5     0D      data    0D = carriage return                          feed
  6     D1      SEP 1   call XMIT
  7     F8      LDI
  8     0A      data    0A = line feed
  9     D1      SEP 1   call XMIT
  A     30      BR
  B     57      addr    reset line position counter and get next data
  C
  D
  E
  F
0070
```

```
R(0)   = Main program counter
R(1)   = XMIT subroutine counter
R(2.0) = XMIT bit counter
R(2.1) = XMIT data storage
R(3)   = XMIT bit timer
R(4)   = HEX-KEY    X register
R(5)   = AUTOMATIC data generator
R(6)   = AUTOMATIC line position counter.
```

ELF II CASSETTE HARDWARE MODIFICATIONS

- by D. Bauer, RR#1, Denmark, Maine, USA 04022

I'd like to pass along a couple of things I've found while working with my ELF II. I have cured my tape loading problems by installing a low pass filter (1.8 K, .01 mfd) in the 'cassette in' line. Also, I made up a box to sit between the computer and the recorder. By using the transformer rather than a series resistor to match the speaker to the recorder output line, I have enough gain to use the speaker as a mike. Now I put voice announcements before each data recording giving the identification, start and stop addresses, date etc. It's crude, but it works. The switch has three positions: Playback, data record and voice record.



ADDENDUM- MANTEI - 2716 Eprom Programmer, IF. No. 23, p. 23.

Be very careful when testing the programmer employing the 'dry run' technique suggested in the article. If the micro is switched off early, in the middle of a 50 ms. WAIT period, the one shot may be continuously triggering. If the 25 v. source is connected to the EPROM during this period, the EPROM could be destroyed. A simple solution would be to modify the program to signal to you when the program is complete, thus preventing such occurances. Change program as follows- at end of program - change 30 1F to 7B 30 20. Now wait until the Q comes on before turning the 1802 off.

MEMORY TEST PROGRAM

- by E.L. Smothers, 5022 July Lynn, Memphis, Tenn., USA 38118

Recently I experienced a lot of troubles with some 21L02's. I used every Test program for memory that I could find. I cleared up some problems, but I still suspected I had memory problems.

So I wrote a little program that I thought would do what I wanted it to do. I worked great for me, maybe it will help someone else.

It checks all 256 byte value combinations for every address starting at 002B. While testing, it displays the HI address under test. If trouble is encountered, Q comes on and the test stops. Pushing the INPUT key displays the LO address. Releasing INPUT allows the test to continue.

After loading the program, flip the RUN switch and go get yourself a cup of coffee. It requires 4 minutes 45 seconds to test 8K using a min 1.758 mhz clock.

MEMORY TEST PROGRAM

PGM to Load at 0000

```
0000  90  A4  B3  B4  B5
  05  F8  2A  A5
  08  F8  2B  A3
  0B  84  53
  0D  E5
  0E  93  55
  10  64
  11  25
  12  E3
  13  84  F3
  15  3A  1D
  17  14
  18  84
  19  32  27
  1B  30  0B
  1D  7B
  1E  83  23  53
  21  3F  21
  23  64
  24  37  24
  26  7A
  27  13
  28  30  0B
```

OR

PGM to Load at any Address

```
0000  F8  00  A4  A5  B3  B4  B5
  07  F8  01  A3  C4
  0B  Same as 000B
```

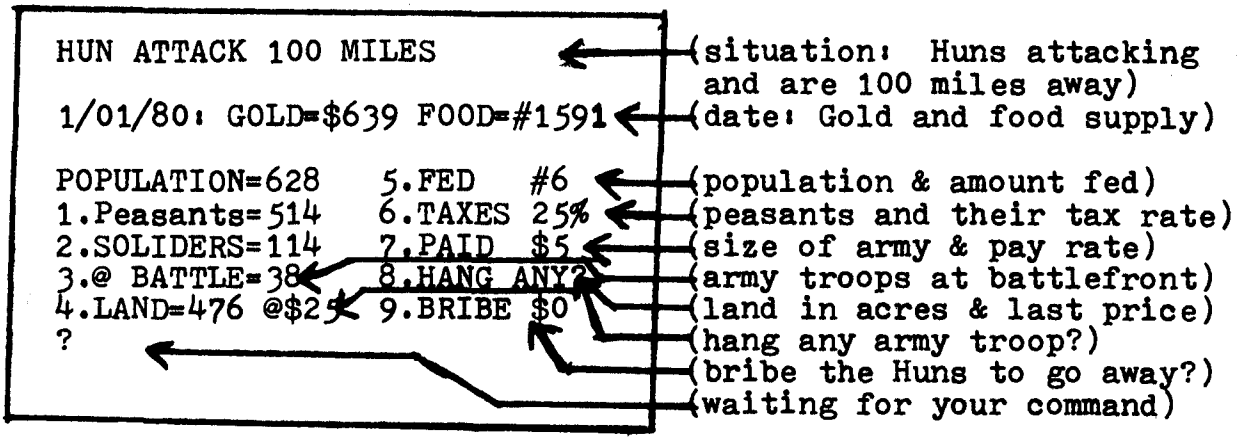
Ed. Note: Program verified on an ELF II with static and dynamic memory.

KINGDOM - A TINY BASIC SIMULATION GAME
- by L. A. Hart

You are the ruler of a tiny kingdom, and are trying to survive despite peasant revolts, army coups, starvation, bankruptcy, or invasion by the Huns. You may even be able to acquire land, get rich, and eliminate poverty and starvation in the kingdom. Your methods can make you another Albert Schweitzer, or make Idi Amin look like a sissy.

Your main resources are land, food, and money. The population is made up of the peasants, who raise food on the land and pay taxes, and the army, which is paid to provide police protection for the peasants and to guard the country against the dreaded Huns. Each month you must decide how best to allocate your resources to keep everybody happy without going broke in the process.

The game starts in January, 1980. A status report is printed, along with any unusual situations, such as riots, Hun attacks, etc. If you are satisfied with things as they are, type"0": If you want to change conditions, type a number (1 thru 9) corresponding to the item you want to affect in the status report. Follow all entries with a carriage return (CR). Several changes can be made each month since the computer will not continue to next month until you type "0". A sample monthly status report might look like this:



The game starts with a randomized set of conditions, but things will not be too unreasonable so no drastic changes should be necessary. The following is a more complete description of what each command does. It can be skipped if you have had previous experience running a country, or just like surprises.

0. No further changes this month: Go on to next month. Taxes are collected and soldiers paid. Food is harvested and distributed.

1. n PROTESTORS: SHOOT ANY? ___ (enter number of peasants to shoot) Your spies have identified "n" peasant protestors. If you want any picked up and shot, enter the number. Note that your spies aren't entirely reliable; they may have no information (? PROTESTORS) or they may not find them all. Also, public executions deter riots, but beware of long-term repercussions!

2. ARMY: +DRAFT, -DISCHARGE? ___ (enter number to be drafted/disch.) A positive number (the + is unnecessary) will draft that many peasants into the army; a negative number will discharge that many soldiers. Note that such transfers are from the police force, not the battlefront.

3. SOLDIERS AT BATTLE=n ? ___ (enter soldiers to be at battlefront) "n" is the number of soldiers currently at the battlefront; to change it, enter the new number desired (the total number). Note that during a battle there are usually casualties, and you can get some idea of the Huns' strength from your casualties.

4. LAND: +BUY, -SELL ? ___ (enter acres of land to be bought/sold) Enter the number of acres to be bought as a positive number (the + is unnecessary), and the number to be sold as negative. The price per acre is that of the last transaction, and prices vary with supply and demand. If you buy/sell large amounts of land it affects the price, usually to your disadvantage. Land is also the obvious source of food, and peasants' home: They like about an acre apiece.

5. PEOPLE FED #n ? ___ (enter new monthly food allowance) Everybody must eat, and "n" determines how much. Allowances of #4 or less should be an act of desperation, since most people dislike starvation. Food production is governed by the amount of land and number of peasants to farm it, weather, time of year (with fall and early winter the best), and mood of the peasants.

6. TAXES n% ? ___ (enter new tax rate)
Enter the tax rate as a number from 0 to 100. At rates above 25% the peasants spend as much time complaining as working, and above 50% you will need a huge police force to keep them from rioting.

7. SOLDIERS PAID $n ? ___ (enter new pay rate for soldiers)
   The army will do your dirty work for you as long as they are paid and fed. You can motivate them to greater efforts with better pay, or by harsh punishment for disloyalty.

8. n PLOT COUP: HANG ANY? ___ (enter number of soldiers to be hung) Loyal officers have reported "n" officers are plotting a coup. Sometimes they don't know who is behind it (? PLOT COUP), and other times they may not identify all of the guilty parties. Naturally, hanging too many or too few have unpleasant results.

9. BRIBE HUNS $? ___ (enter number of dollars to bribe Huns)
   The Huns can be bribed to go away and bother somebody else. Enter the number of dollars only, not the $ sign. Note that once bribed, they will demand that amount every month as a tribute to their obvious superiority.

There are two more commands not shown on the status report, but nevertheless useful at times:

10. I give up! Let me out before I get lynched!
    This command will return you to BASIC. To play again, type "RUN".

11. Reprint the status report for the current month.
    This command is useful to see what effect a long list of previous commands has had, or to check conditions in case you've forgotten.

The game proceeds from month to month. Your goal is to see how long you can stay in power without losing your throne. In the event you make a fatal mistake, the message "YOU LOSE!" will be printed after the condition that cause your defeat. Specifically, you lose if you go broke, run out of food, let the Huns get to the capitol, have a successful peasant revolt, or a military coup.

A few words of advice: The game starts fairly well balanced, so don't make drastic changes until you know what you're doing. Both the peasants and the army hold a grudge against mistreatment; you may crush them now, but they'll get you later if given a chance. Happy peasants work harder than unhappy ones. Also, don't expect instant results; some things take time.

Don't be afraid to try different approaches to running the country. There is no right way to do it, and you can succeed with a harsh militaristic state or a benevalent, peaceful one if you do it right.

The program is less than 2K long in tiny BASIC, so a number of compromises were made to make it fit (with 2K for Tiny BASIC itself) into a 4K system. Messages are very short and brief. Operator inputs are only checked for obvious errors, and it is easy to type a ridiculous entry and bomb out of the game, or get bizzare results. (For example, nothing prevents you from hanging your entire population). These problems can easily be fixed with more memory. Have fun!

```
IDIOT 1.1
*$P3

:REM LISTING OF KINGDOM
:REM KINGDOM
:REM BY LEE A. HART
:LIST

1 A=RND(Ø,99)
2 B=Ø
3 C=Ø
4 D=RND(99,999)
5 E=4ØØ
6 F=6
7 L=E+2*A
8 H=149Ø
11 K=25
12 G=D+L*2
13 M=Ø
16 P=L+A
18 R=Ø
19 S=2*A
2Ø T=25
21 U=1ØØ
22 V=Ø
23 W=5
24 X=Ø
25 Y=79
26 Z=Ø
2ØØ V=V+RND(Ø,5)+(L/4Ø*K/9+D/15+999/U-2*A-B/2+R)/9
2Ø5 IFV<-9V=Ø
21Ø IFV>ØPR"HUN ATTACK ";U;" MILES"
215 IFU<1GOTOH
24Ø R=(R-W*F)/2+Z+12
25Ø IFR>ØPR"ARMY PLOTS COUP"
255 IFR>1C=C+R
257 IFC>5ØGOTOH
26Ø Z=Z+RND(Ø,2)+P/L+75/(1Ø1-T)-S*21/P+5/F
27Ø IFZ>ØPR"PEASANT RIOTS"
29Ø IFC>5ØGOTOH
295 IFZ<-3Z=-3
```

```
300 M=M+1
310 IFM>12M=1
320 IFM=1Y=Y+1
325 PR
330 PRM;"/1/";Y;": GOLD=$";D;" FOOD=#";G
333 IFG<0GOTOH
335 IFD<0GOTOH
338 PR
340 PR"POPULATION=";A+P+S;"  5.FED  #";F
350 PR"1.PEASANTS=";P;"  6.TAXES ";T;"%"
351 PR"2.SOLDIERS=";A+S;"  7.PAID $";W
355 PR"3.@ BATTLE=";A," 8.HANG ANY?"
370 PR"4.LAND=";L;" @$";K;" 9.BRIBE $";B
400 INQ
420 IFQ>10GOTO330
430 IFQ<0GOTOE
440 GOTOQ*100+500
500 G=G+L/4/(L*9/P+P*9/L)*M*(RND(0,4)-Z)-(A+P+S)/19F
515 IFF>3GOTO550
520 PRP/9/F;" PEOPLE STARVED"
530 P=P-P/9/F
550 IFV>0A=A-V
560 U=U-V+A
565 IFU>99U=100
570 IFA<1A=0
575 K=K+RND(-1,1)
580 D=D-(A+S)*W+P/20*T
590 P=P+RND(0,9)
595 GOTO200
600 IFZ>0PR" ";Z+RND(0,4);
610 IFZ<1PR" ?";
620 PR" PROTESTORS: SHOOT ANY";
630 INQ
640 Z=Z-Q
650 P=P-Q
660 GOTOE
700 PR" ARMY: +DRAFT,-DISCHARGE";
710 INQ
720 IFQ>PQ=P
725 IFQ<-SQ=-S
730 P=P-Q
740 S=S+Q
750 GOTOE
800 S=S+A
810 PR" SOLDIERS @ BATTLE=";A,
820 INA
825 IFA<0A=0
830 IFA>SA=S
840 S=S-A
850 GOTOE
900 PR" LAND;+BUY,-SELL",
910 INQ
920 L=L+Q
930 D=D-Q*(K+Q/5)
940 K=K+Q/10
950 GOTOE
```

```
1000 PR" PEOPLE FED  #";F,
1010 INF
1015 IFF<1F=0
1020 GOTOE
1100 PR" TAXES ";T;"%",
1110 INT
1120 IFT>99T=100
1130 GOTOE
1200 PR" SOLDIERS PAID ";W,
1210 INW
1220 IFW<0W=0
1230 GOTOE
1300 IFR>0PR" ";R;
1310 IFR>1PR" ?";
1320 PR" PLOT COUP: HANG ANY";
1330 INQ
1340 IFQ>SQ=S
1350 S=S-Q
1370 R=R-Q
1380 IFQ>1C=C+1
1390 GOTOE
1400 PR" BRIBE HUNS $",
1410 INB
1420 IFB<0B=0
1430 GOTOE
1490 PR"YOU LOSE!"
1500 END
```

QUEST BASIC PRINTER ROUTINE

Quest basic version 5 has a small printer routine ,which can be accessed with a POUT instruction.

When I tried to use the printer as an output device every thing was dumped in elongated characters .The Centronics 730 as well as other Centronics printers that have elongated character capability will do so if bit 7 (MSB) is 1.

In version 5 of Quest basic the printer routine is located at 06D3 hex.Most of the printers also have an auto linefeed,so while we are at it we can trap the linefeeds.

06DD3 22 9F 52 FB 0A 32 DE 36 DA 63 D5 12 D5

The elongated characters can still be used in Basic by using the following sequence in your PRINT statement.....

ESC CTRL N to start,and ESC CTRL O to end the elongations. This sequence was used to print the header of this article.

The listing of the Basic text editor which I used,was originaly supplied with Quest's Basic version 1.4 manuel and modified to include a PRINT statement to dump the text to the printer.

A very good use for the editor would be to make up some articles for IPSO FACTO.....

Good luck !!!

```
0 REM.TEXT EDITOR,TAPE 1 SIDE A
10 DEFINT Z
20 CLS: PRINT "*****THIS IS A BASIC EDITOR*****"
40 PRINT "ENTER--LIST--EDIT--PRINT--TAPE--BYE"
70 INPUT A$
80 CLS
90 IF MID$(A$,1,2)="EN" GOTO 140
100 IF MID$(A$,1,2)="LI" GOTO 300
110 IF MID$(A$,1,2)="ED" GOTO 400
112 IF MID$(A$,1,2)="TA" GOTO 250
113 IF MID$(A$,1,2)="PR" GOTO 800
115 IF MID$(A$,1,2)="BY" GOTO 10000
120 PRINT "CAN'T UNDERSTAND---TRY AGAIN"
130 WAIT(300): GOTO 40
140 INPUT "DATA FROM TAPE OR KEYBOARD"Z$
142 IF MID$(Z$,1,2)="TA" INPUT Z$: DLOAD C: GOTO 40
145 IF MID$(Z$,1,2)="KE" GOTO 150
146 GOTO 120
150 INPUT "HOW MANY LINES ON THE PAGE"B
152 INPUT "START WITH LINE #"S
154 INPUT "MAX CHAR/LINE"C
156 PRINT "TYPE (DONE) TO EXIT THE ENTER MODE"
160 PRINT "READY TO ACCEPT ";B;" LINES OF ";C;" CHAR.OR LESS"
162 M=0
165 IF S>B PRINT "LINE # OUT OF RANGE": GOTO 140
170 FOR A=S TO B
180 IF M=1A$(A)=" ": GOTO 210
190 INPUT A$(A)
200 IF A$(A)="DONE"M=1:A$(A)=""
205 IF C<LEN(A$(A)) PRINT "LINE TOO LONG": GOTO 190
210 NEXT
220 PRINT "PAGE COMPLETED": GOTO 40
250 INPUT "PLACE  RECORDER IN THE RECORD MODE AND PRESS RETURN"Z$
260 DSAVE C: GOTO 40
300 PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT
310 FOR A=1 TO B
320 PRINT A$(A)
330 NEXT
340 PRINT : PRINT : PRINT : PRINT : PRINT : PRINT : PRINT
350 GOTO 40
400 PRINT "BASIC IS NOW IN THE EDIT MODE"
402 INPUT "DO YOU WANT A NUMBERED LISTING?"Z$
404 IF MID$(Z$,1,2)="Y" GOSUB 600
410 INPUT "WHICH LINE IS TO BE EDITED?"L
425 IF L=0 GOTO 40
430 PRINT A$(L)
440 INPUT "SHOULD IT BE CHANGED?"Z$
460 IF MID$(Z$,1,1)="Y" GOTO 480
470 GOTO 410
480 INPUT A$(L)
490 GOTO 410
600 FOR A=1 TO B
610 PRINT A;TAB(3);A$(A)
620 NEXT
630 RETURN
800 POUT
803 PRINT
805 FOR A=1 TO B
810 PRINT TAB(10);A$(A)
820 NEXT
830 PRINT
890 TOUT
900 GOTO 40
10000 CLS: END
```

RATRACE 1 - A SIMULATION GAME IN QUEST SUPER BASIC VER 3.0

- by P. B. Liescheski III, 4510 Duval St., #203, Austin, Tx, USA 78751

Ed. Note: Program is designed to run with Quest Basic V3.0 and a memory map video display Bd VBIB. The Program must be modified to work with other devices.

RATRACE1 is a program written in Quest Super BASIC V3.0 which simulates the running of a rat maze. The program is based on a simple optimized right-hand algorithm. The article, "Simple Maze Traversal Algorithm", by Sandra and Stephen A. Allen in BYTE has a detailed description and development of this and other maze traversal algorithms (BYTE, June 1979, p.36).

Basically the program simulates the run using the simple right-hand algorithm. Each move is stored on stack. The maze grid matrix which contains information on the structure of the maze and the previous moves of the simulated rat, is checked in order to determine whether the rat is crossing its tracks. If it realizes that it is running in circles or backtracking, it merely removes that portion of the run from stack and thus optimizes the run. After optimization, the computer simulates a second run based on the moves remaining on the move-stack.

The matrix M is used as the maze grid matrix. The value of each M matrix element designates the status of that portion of the maze. The value 80 represents an allowable path in the maze; the value 81 represents a path which has been crossed by the rat, while the value 0 represents an obstacle in the maze. The matrix R is used as the stack in which the moves of the rat are stored. Since the rat is moving in a particular direction, two points are used to define the position and direction of the simulated rat. These points are contained in vectors O and N. The vector O contains the old or previous position of the rat, while N contains the new position of the rat. In other words, O defines the back of the rat while N defines its front.

In an actual run of the program, the structure of the maze is first entered. The program assumes that the maze can be defined within a 10*10 matrix with elements (1,1) as the starting point and (10,10) as the goal. The maze is entered as a string of asterisks and blanks. This string must be 10 characters in length or errors may result. The asterisks denote an allowable path, while the blanks denote an obstruction in the maze. After the input, the maze grid is displayed, and the starting position of the rat is set. With this, it increments the move counter M (a scalar variable) and attempts to make a right turn. If it cannot make a right turn, then it attempts to go forward. If it cannot go forward, it attempts to make a left turn or finally it backtracks. After the move, the coordinates are stored in the move-stack R, while the value of the new position in the maze grid matrix M is examined. If its value is 81, which indicates that it has been previously at this position, then the move counter M which also serves as the move-stack pointer, is decremented until these coordinates are again found in the stack. In other words, the stack is optimized by removing the inefficient moves. After this, it checks to see whether it is finished and then proceeds to the next move.

RATRACE1 has been written for the VB1B video display board. It should also work with the Gremlin video board, but with the following general display modifications:

```
POKE(C + @E08B + 32*(R-1), #XX)
```

This program will not work very well with a printer. Other video memory boards should work with proper modifications to the output POKE statements. Also the

program is designed to perform an optimized right-hand run, but by interchanging the instructions in lines 320 and 380, the program will perform the run according to an optimized left-hand algorithm. It is the author's intent to develop another program RATRACE2 which will run a maze according to an optimized run that is based on a combined right- and left-hand algorithm.

Finally, it should be noted that the listing and the example input presented in this paper were prepared by a VAX system, since the 1802 system does not have a printer. For this reason, the four-digit numbers in the extreme left-hand portion of the page have no significance to the 1802 BASIC program. They are merely the results of the VAX SOS editor.

```
 100
 200
 300    ENTER MAZE:
 400              1234567890
 500    ROW #1    ?***  ****
 600    ROW #2    ?* ****   *
 700    ROW #3    ?* *   *
 800    ROW #4    ?*********
 900    ROW #5    ?  *      *
1000    ROW #6    ?* * *****
1100    ROW #7    ?*** * * *
1200    ROW #8    ?   ** * *
1300    ROW #9    ?   * * * *
1400    ROW #10   ?   **** **
1500
1600
```

```
   5  REM **
  10 REM ** RATRACE 1
  20 REM ** Phillip B. Liescheski III * 6-29-81
  25 REM **
  30 REM ** An optimized right-hand maze algorithm
  35 REM **
  37 DEFINT Z
  40 DIM O(2),N(2),M(10,10),R(2,50)
  43 REM **
  45 REM ** Input maze grid matrix
  47 REM **
  50 PR "ENTER MAZE:":PR TAB(11);"1234567890"
  60 FOR R=1 TO 10
  70 PR "ROW #";R;TAB(10);
  80 INPUT A$
  90 FOR C=1 TO 10
 100 M(R,C)=0
 110 IF MID$(A$,C,1)="*" THEN M(R,C)=80
 120 NEXT C
 130 NEXT R
 135 REM **
 140 REM ** Display maze
 145 REM **
 150 CLS
 160 FOR R=1 TO 10
 170 FOR C=1 TO 10
 175 IF M(R,C)=0 GOTO 190
 180 POKE(C+@F08B+64*(R-1),$2A)
 190 NEXT C
 200 NEXT R
 205 REM **
 210 REM ** Perform first right-hand run
 215 REM **
 220 REM * Initial move
 225 REM *
 230 M=0:F=0
 240 O(1)=0:O(2)=1
 250 N(1)=1:N(2)=1
 255 REM **
```

```
260 REM ** Display the move
265 REM **
270 GOSUB 1000
275 F=1
278 REM **
280 REM ** Check if finished
285 REM **
290 IF N(1)=10 IF N(2)=10 GOTO 570
293 REM *
295 REM * Bump move counter/stack pointer
300 M=M+1
305 REM *
310 REM * Attempt a right-hand turn
320 GOSUB 1090
330 IF X>0 IF X<11 IF Y>0 IF Y<11 IF M(Y,X)>=80 GOTO 430
325 REM *
340 REM * If not, try forward
350 GOSUB 1130
360 IF X>0 IF X<11 IF Y>0 IF Y<11 IF M(Y,X)>=80 GOTO 430
365 REM *
370 REM * If not, try a left-turn
380 GOSUB 1050
390 IF X>0 IF X<11 IF Y>0 IF Y<11 IF M(Y,X)>=80 GOTO 430
395 REM *
400 REM * If not, backtrack
410 GOSUB 1170
415 REM *
420 REM * Update old/new move coordinates
430 O(1)=N(1):O(2)=N(2)
440 N(1)=X:N(2)=Y
445 REM *
450 REM * Push next move coordinates on move-stack
460 R(1,M)=X:R(2,M)=Y
465 REM *
470 REM * Check if going in circles
480 IF M(Y,X)<>81 GOTO 530
485 REM *
490 REM * If so, optimize move-stack
500 M=M-1
510 IF R(1,M)=X IF R(2,M)=Y GOTO 530
520 GOTO 500
525 REM *
530 REM * Mark its tracks on maze grid
540 M(Y,X)=81
545 REM *
550 REM * Next move
560 GOTO 270
565 REM **
570 REM ** Perform right-hand optimized run
575 REM * Clean-up maze on display
578 REM *
580 POKE(10+@F08B+64*(10-1),#2A)
585 POKE(1+@F08B,#52)
588 REM *
590 REM * Initialize move counter
600 M=1:F=0
603 REM **
605 REM ** Display optimized run
608 REM **
610 N(1)=R(1,M):N(2)=R(2,M)
620 GOSUB 1000
630 IF N(1)=10 IF N(2)=10 END
640 M=M+1
650 GOTO 610
660 END
1000 REM **
1005 REM ** Move display subroutine
1008 REM **
1010 IF F=0 GOTO 1030
1020 POKE(O(1)+@F08B+64*(O(2)-1),#2A)
1030 POKE(N(1)+@F08B+64*(N(2)-1),#52)
1040 RETURN
1045 REM **
1050 REM ** Left-turn subroutine
1055 REM **
1060 X=N(1)+(N(2)-O(2))
1070 Y=N(2)+(O(1)-N(1))
1080 RETURN
1085 REM **
1090 REM ** Right-turn subroutine
1095 REM **
1100 X=N(1)+(O(2)-N(2))
1110 Y=N(2)+(N(1)-O(1))
1120 RETURN
1125 REM **
1130 REM ** Forward move subroutine
1135 REM **
1140 X=N(1)+(N(1)-O(1))
1150 Y=N(2)+(N(2)-O(2))
1160 RETURN
1165 REM **
1170 REM ** Backtrack subroutine
1175 REM **
1180 X=O(1)
1190 Y=O(2)
1200 RETURN
9999 END
```

1802 SERIAL I/O BOARD

- by T. Crawford, 50 Brentwood Dr., Stoney Creek, Ont. L8G 2W8

I - HARDWARE:

A) General Description

This board provides 3 hardware US/ART's for serial I/O communications. The US/ART's are mapped into memory space for communication with the processor. The interface to the outsideworld is through RS-232 drivers/receivers.

The baud rate for each US/ART is individually programmable under software control. This is accomplished using a Counter-Timer Chip (CTC), which is also memory-mapped for processor access. All US/ART and CTC Functions are under complete software control.

The CTC and the 3 US/ART's occupy 16 bytes of memory, located at any 16 byte boundary in the I/O page. Only 10 of the 16 bytes are used; 2 for each of the US/ART's and 4 for the CTC. Three interrupt lines are provided from the serial I/O board; 1 for each of the US/ART's. These are accessed via the edge connector on the outer end of the board, and must be wired to an interrupt-interface board if interrupt-mode operation is desired.

The CTC uses the CPU block frequency as a basis for generating the US/ART baud rates. It is best if this is crystal-controlled to ensure stable, accurate baud rates. If the CPU clock frequency exceeds 2.0 MHz, then the CTC can be driven from TPA, by cutting a trace and installing a jumper on the board near the system edge connector. Since the board has spare RS232 drivers and receivers, it is possible to buffer Q and one or more EF lines by installing some jumpers. This allows easy use of a software UART function as well as the on-board hardware US/ART's. A buffered Q signal is brought out to the I/O edge connector, as well.

Mode Instruction Definition

Figure 1 shows the format of the Mode Instruction for Asynchronous Mode. Figure 2 shows a definition of Asynchronous Mode.

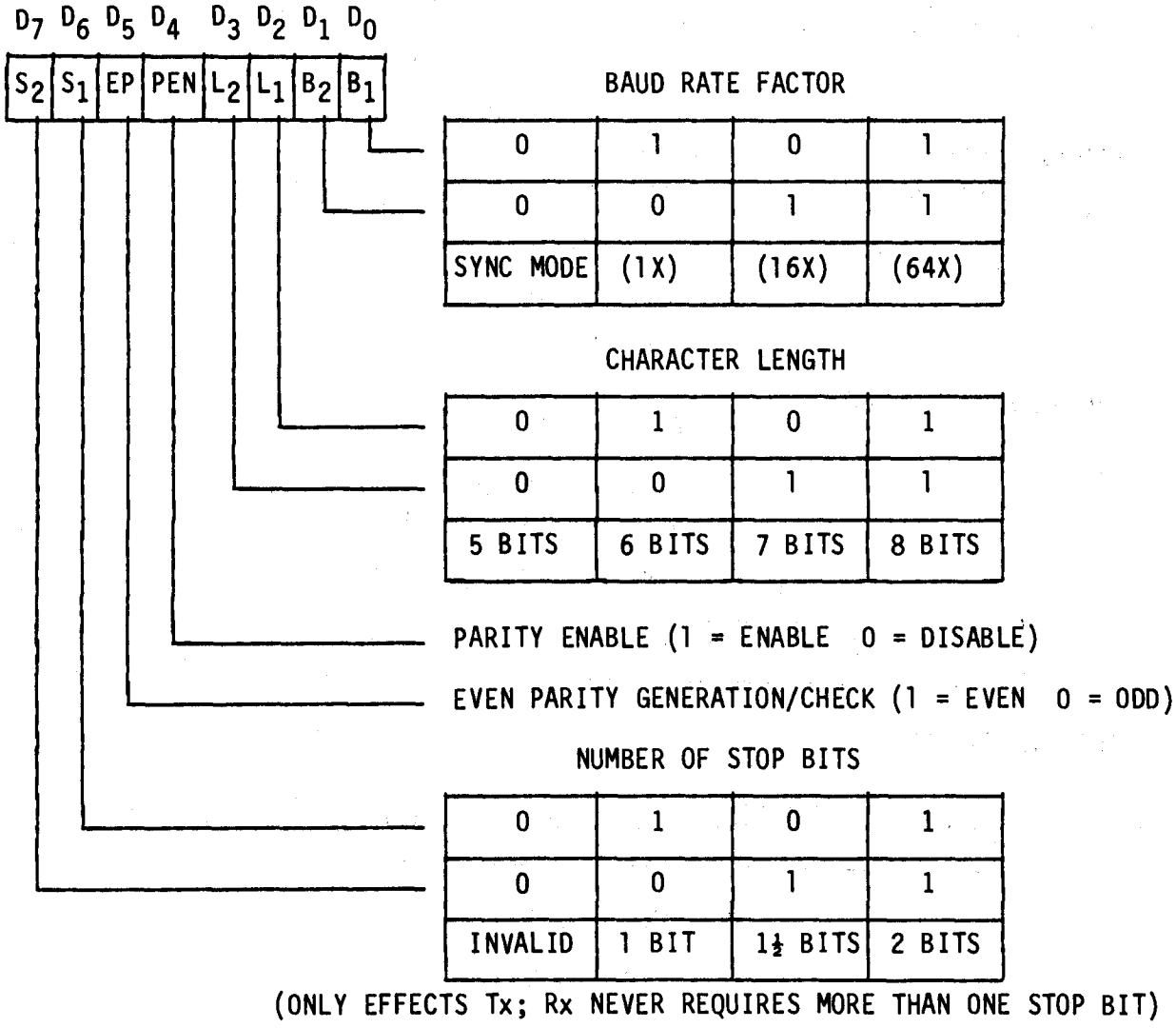
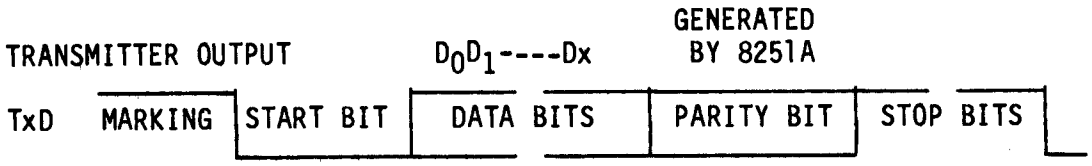
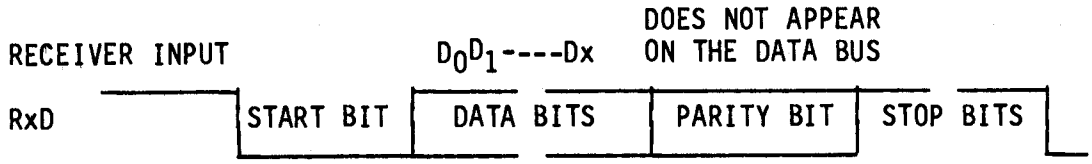


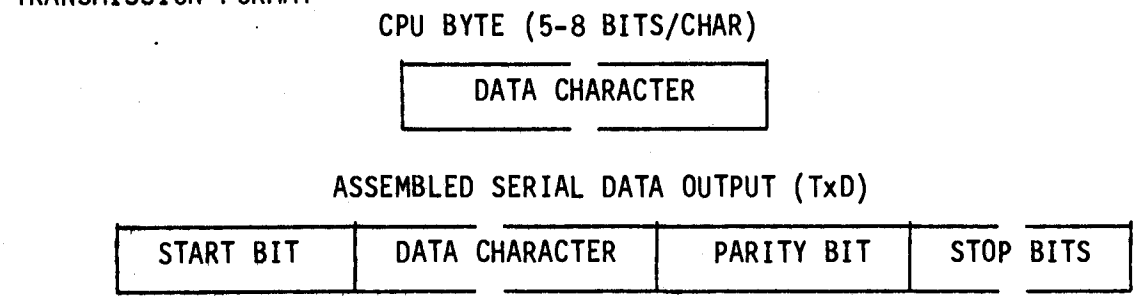
Figure 1: Mode Instruction Format, Asynchronous Mode



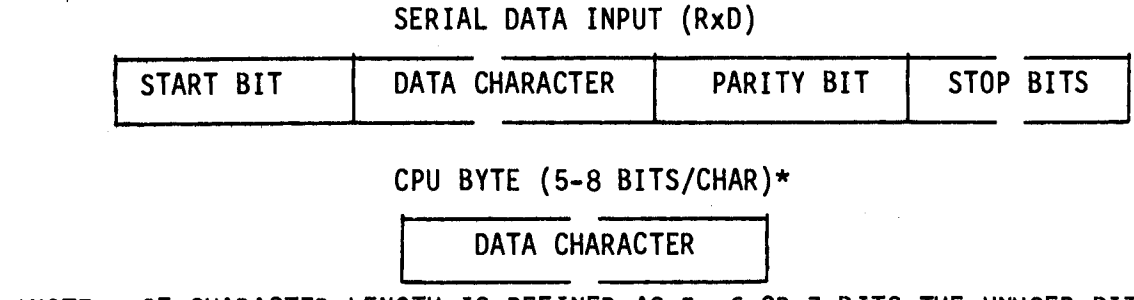


PROGRAMMED CHARACTER LENGTH

TRANSMISSION FORMAT



RECEIVE FORMAT



*NOTE: IF CHARACTER LENGTH IS DEFINED AS 5, 6 OR 7 BITS THE UNUSED BITS ARE SET TO "ZERO"

Figure 2: Definition of Asynchronous Mode

NOTE: When parity is enabled it is not considered as one of the data bits for the purpose of programming the word length. The actual parity bit received on the Rx Data line cannot be read on the Data Bus. In the case of a programed character length of less than 8 bits, the least significant Data Bus bits will hold the data; unused bits are "don't care" when writing data to the 8251A, and will be "zeros" when reading the data from the 8251A.

COMMAND INSTRUCTION DEFINITION

Once the functional definition of the 8251A has been programmed by the Mode Instruction, the device is ready to be used for data communication. The Command Instruction controls the actual operation of the selected format. Functions such as: Enable Transmit/Receive, Error Reset and Modern Controls are provided by the Command Instruction.

Once the Mode Instruction has been written into the 8251A, then all further "control writes" (C/D=1) will load a Command Instruction. A Reset Operation (internal or external) will return the 8251A to the Mode Instruction format.

Figure 3 shows the Command Instruction Format.

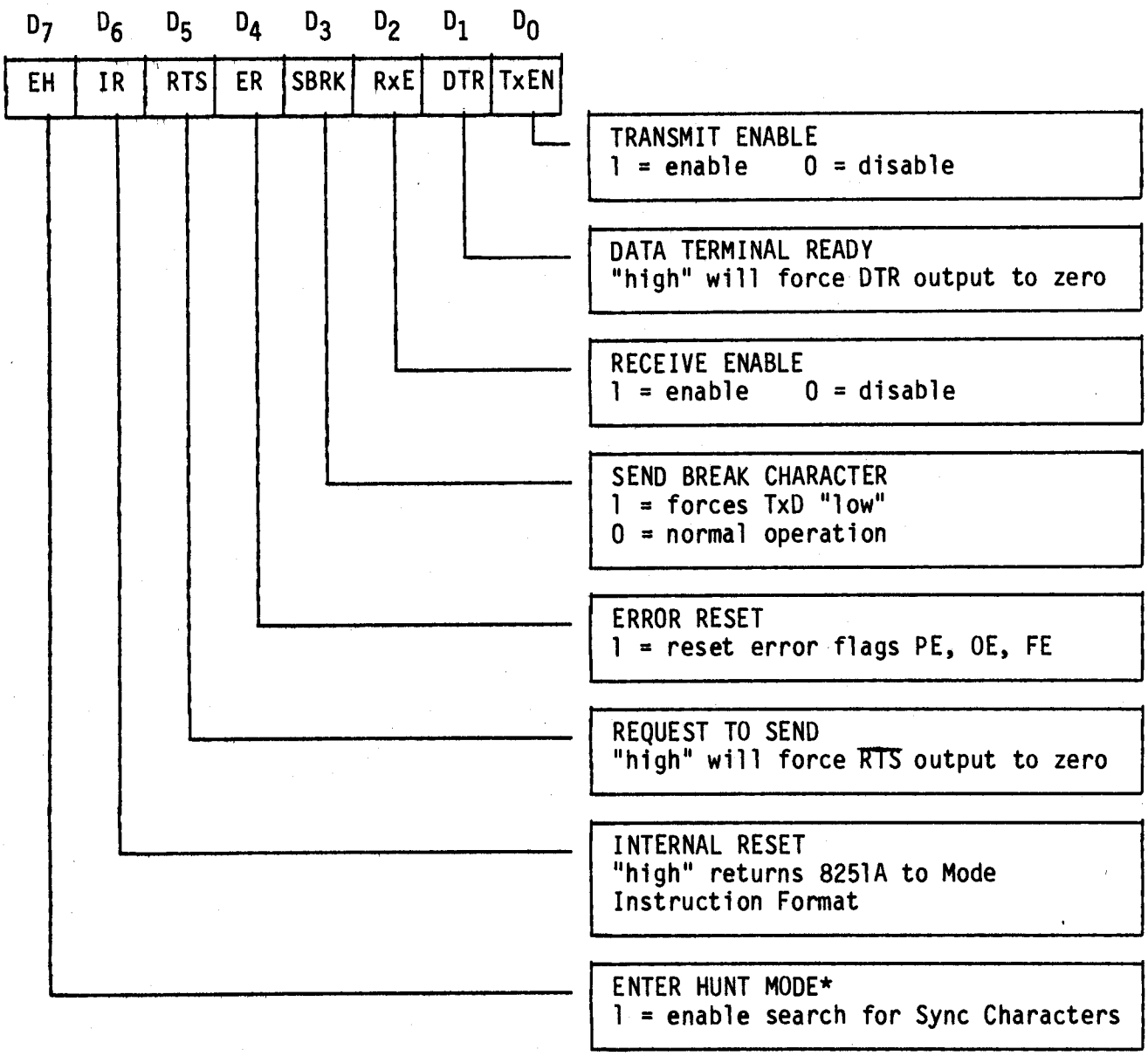
STATUS READ DEFINITION

In the data communication systems it is often necessary to examine the "status" of the active device to ascertain if errors have occurred or other conditions that require the processor's attention. The 8251A has facilities that allow the programmer to "read" the status of the device at any time during the functional operation. (The status update is inhibited during status read).

A normal "read" command is issued by the CPU with C/D=1 to accomplish this function.

Some of the bits in the Status Read Format have identical meanings to external output pins so that the 8251A can be used in a completely Polled environment or in an interrupt driven environment. TxRDY is an exception.

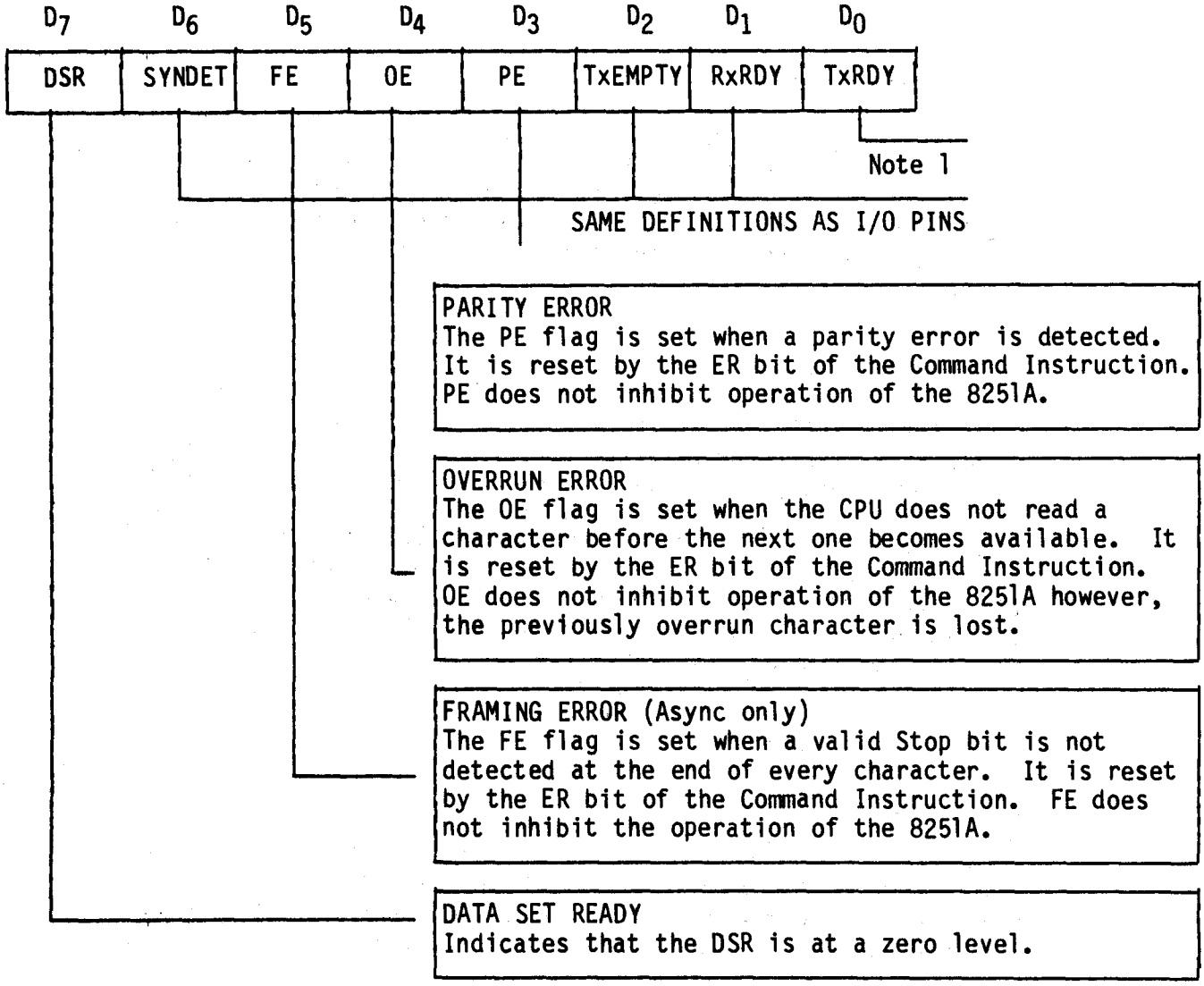
Figure 4 shows the Status Read Format



* (HAS NO EFFECT IN ASYNC MODE)

Note: Error Reset must be performed whenever RxEnable and Enter Hunt are programmed.

Figure 3: Command Instruction Format



Note 1: TxRDY status bit has different meanings from the TxRDY output pin. The former is not conditioned by CTS and TxEN; the latter is conditioned by both CTS and TxEN.
i.e. TxRDY status bit = DB Buffer Empty
TxRDY pin out = DB Buffer Empty (CTS=0) (TxEN=1)

Figure 4: Status Read Format

II - SOFTWARE:

A) Introduction

Before the Serial I/O board can be used, the CTC registers and US/ART control ports must be set up. This can most conveniently be done as part of the micro-computer initialization logic. If the settings must be changed after initialization, utility routines can be written for this purpose.

B) US/ART Initialization

The Intel 8251A US/ART chip can be used for either asynchronous or synchronrous data communication. The following description applies to its use for asynchronous communication only. For details concerning synchronous operation, and for a very detailed description of the 8251A, the reader is referred to the Intel Component Data Catalog, 1978, Section 12 (MPU Peripherals). The control words which must be sent to the 8251A immediately after reset (external or internal) are split into two formats:

1. Mode instruction,
2. Command Instruction

Mode Instruction

This format defines the general operational characteristics of the 8251A. It must follow a Reset operation (internal or external). Once the Mode Instruction has been written into the 8251A by the CPU, Command Instructions may be inserted.

Command Instruction

This format defines a status word that is used to control the actual operation of the 8251A. Both the Mode and Command Instructions must conform to a specified sequence for proper device operation. The Mode Instruction must be inserted immediately following a Reset operation, prior to using the 8251A for data communication. All control words written into the 8251A after the Mode Instruction will load the Command Instruction. Command Instructions can be written into the 8251A at any time during the operation of the 8251A. To return to the Mode Instruction format, the master Reset bit in the Command Instruction word can be set to initiate an internal Reset operation which automatically places the 8251A back into the Mode Instruction format. Command Instructions must follow the Mode Instructions.

C) CTC INITIALIZATION

The 8253 programmable counter/timer chip is organized as 3 independent 16 bit counters. The chip itself is seen by the CPU as a set of 4 contiguous I/O bytes; three are counters and the fourth is a control register for Mode programming. (see Figure 5.).

The CTC can be programmed to operate in any one of 6 Modes. For baud rate generation, Mode 3, Square Wave Rate Generation, is used. This implements a free-running divide-by-N counter, where output of the counter remains high until one half the count has been completed (for even numbers) and goes low for the other half of the count. If the count is odd, the output will remain high for (N+1)/2 counts, and low for (N-1)/2 counts. All 3 counters are supplied with the CPU clock frequeny from the buss at their clock inputs, typically 1.78977 MHz.

Initialization software must program each counter with the mode and count value desired. This involves writing a Mode control word and the programmed number of count register bytes (1 or 2) for each of the 3 counters.

The Mode control register has the format shown in Figure 6. For baud rate purposes, each counter will be programmed to accept 2 count value bytes and will operate in binary. Table 1 provides count values for a range of standard baud rates.

Note that the 3 bytes written to the CTC to initialize each counter cannot be separated by any other access to or from that counter. The result of this is that the memory change (M) function of the RCABUG monitor cannot be used, when testing, to initialize a CTC counter, since it does a read-after-write, to confirm a memory change. The M function CAN be used once the CTC is initialized, and for any USA/RT access.

(Ø=1.78977MHz)

Baud Rate	16X Clock (Hz)	Count Value (Dec)	(Hi)	(Lo)	Actual Rate (Hz)	Baud Rate Error (%)
75	1200	1491	05	D3	1200	0
110	1760	1017	03	F9	1760	0
300	4800	373	01	75	4798	-0.04%
600	9600	186	00	BR	9622	+0.23%
1200	19200	93	00	5D	19245	+0.23%
2400	38400	47	00	2F	38080	-0.83%
4800	76800	23	00	17	77816	+1.32%
9600	153600	12	00	0C	149145	-2.90%
19200	307200	6	00	06	298295	-2.90%

TABLE 1: CTC Register Values for Various Baud Rates

TYPICAL INITIALIZATION CODE

This code provides logic to set up the Serial I/O Board as follows:

a) All US/ARTS set up for asynchronous mode, 7 data bits, even parity, 1 stop bit, 16x clock, DTR true, RTS true.

b) All 3 CTC channels set up to generate a 1200 baud rate (generated rate - 19200 Hz).

c) Assume the Serial I/O board is mapped into the first 16 bytes of the memory-mapped I/O page (#FF00 to #FF01).

```
RMMIO   EQL   RE        ; Assign a memory-mapped I/O register.
        LDI   #FF       ; Set up hi byte of I/O page pointer.
        PHI   RMMI0
        LDI   #0B       ; Point to CTC control register.
        PLO   RMMI0
        LDI   #36       ; Program mode 3 for
        STR   RMMI0     ; Counter #0,
        LDI   #76
        STR   RMMI0     ; Counter #1,
        LDI   #B6
        STR   RMMI0     ; And Counter #3.
        LDI   #5D       ; Get low byte of count value,
        DEM   RMMI0     ; Point to Counter #2,
        STR   RMMI0     ; And store the lo byte.
        DEC   RMMI0
        STR   RMMI0     ; Similarly for Counter #1,
        DEC   RMMI0
        STR   RMMI0     ; And Counter #0
        LDI   #0A       ; Point to Counter #2 again,
        PLO   RMMI0
        LDI   #00       ; Get hi byte of count value,
        STR   RMMI0     ; and store it.
        DEC   RMMI0
        STR   RMMI0     ; Similarly for Counter #31,
        DEC   RMMI0
        STR   RMMI0     ; And Counter #0,

                CTC Setup is Complete.

        LDI   #01       ; Point to US/ART#0 control register,and
        PLO   RMMIO
        LDI   #7A       ; set up Mode Instruction of US/ART #0.
        STR   RMMIO
        LDI   #37       ; set up Command Instruction of US/ART#0
                          and reset error flags.
        STR   RMMI0
        LDI   #27
        STR   RMMI0     ; Error Flags are now reset.
        INC   RMMI0     ; Point to next US/ART
        INC   RMMI0
        LDI   #7A       ; Set up Mode Inst. of US/ART #1,
        STR   RMMI0
        LDI   #37       ; and Command Inst.,
        STR   RMMI0     ; and reset error flags.
        LDI   #27
        STR   RMMI0
        INC   RMMI0     ; Point to next US/ART
        INC   RMMI0
        LDI   #7A       ; Set up mode Inst. of US/ART #2,
        STR   RMMI0
        LDI   #37       ; and Command Inst.,
        STR   RMMI0     ; and reset error flags.
        LDI   #27
        STR   RMMI0     ; US/ARTS are now set up.

;

;               SERIAL I/O Board setup complete.
;
```

III SERIAL I/O BOARD TESTING

A) BUSS INTERFACE CHECKS

The first stage in testing the completed Serial I/O Board is to check out the address decoding and control buffering logic. This involves installing only IC numbers 1,2,3,4,7, and 8 on the board, then plugging it into the system bus. Using variants of the program loop in Listing 1, exercise the board and test for proper board select (IC3, pin 8) and chip select signals (IC7, pins 1 to 7, and 9), using a logic probe, or preferably an oscilloscope. Try different settings of the address select switches on the board, in combination with different lo-byte values in RE. Check that the control signals (MRD, MWR, AO, TPB, and CTC-WR) all appear correctly.

Now install IC's 5 and 6, the data bus buffers and repeat the above checkout, but using the code in Listing 2. In addition, check the data lines on the board, between the data bus buffers and the US/ART's, for the data byte being written to the board. (Note that this check will be very difficult to do without a 2 channel oscilloscope. Skip it if you don't have one). Do this check using different data bytes, so as to check for both O's and 1's on all 8 data lines.

Now change back to the code in listing 1, and look for a 00 data byte being read from the serial I/O board. Locate the 8 data bus pull-down resistors (about halfway along the right hand side, when holding the board component side up, buss edge connectors closest to you). Using a jumper wire in series with a 1K ohm resistor connected to + 5V, pull up each of the data bus lines at the pull-down resistors (be sure to contact the data end, not the grounded end of each resistor!). Check that the appropriate data byte can be read from the Serial I/O board.

B) CTC CHECKS

Once all of the above checks have been successfully made, install the 8253 (IC9) into the board. Load the initialization code presented previously, and execute it. Using a scope or frequency counter, you should now be able to measure a 19200 Hz.TTL square wave at pins 10,13, and 17 of the 8253 CTC. If not, then I suggest that all of the checks made using Listing I and 2 be repeated, but leave IC's 5,6 and 9 installed. Examine all waveforms at the CTC pins for correct levels and timing. Note that an oscilloscope is mandatory at this point. If all signals prove correct, the CTC is probably bad, and should be replaced. Do not proceed with the US/ART checkout until the CTC is working properly. Once you have the CTC working, you may want to become more familiar with its operation by trying out different time constant values and control register values in the initialization code.

C) US/ART CHECKS

After successfully checking out the CTC, power down and install one of the 8251 US/ART's into ICI2 socket, and install IC's 18 and 19. Connect an RS232 interfaced printer or CRT terminal to pins 25 (RxD) and 6 (TxD) of the I/O edge connector. Connect pin 26 (DSR) and pin 27 (CTS) to pin 4 (DTR). Now ensure that the terminal is ready to receive data at 1200 baud, 7 data bits, even parity and 1 stop bit. If this is not possible, then change the time constant value for Counter #0 in the initialization code to a suitable value to match the terminal (See table 1). Also, change the Mode Instruction for US/ART#0 to be compatible with the terminal (see Figure #1).

Load the initialization code and execute it. Check for the proper baud rate at the US/ART, pins 9 and 25. Check for proper RS232 levels at I/O edge connector, pins 6,4,5,25,26 and 27.

Load the code in Listing 3 and execute it. This should produce a continuous string of ASCII "2"'s on the terminal, until the CPU is reset. Note that after resetting the CPU, the initialization code must be executed again.

The final test, using Listing 4, checks the serial input part of the US/ART. The routine simply waits for a received character, then reads it and sends it back out to the US/ART, to be displayed at the terminal. Once this test is passed, the remaining US/ART's and RS232 interface IC's (14,15,16, and 17) can be installed and checked out, using a procedure similar to that used for US/ART #0. Remember to use the correct I/O connector pins and addresses for each US/ART.

D) INTERRUPT MODE CHECKS

The final IC, 13, can be installed or left out, as desired. When installed, it simply OR's 3 signals from each US/ART, to produce an interrupt signal. This results in 1 interrupt signal per US/ART, available on the I/O edge connector. Interrupt software responding to these signals must read the US/ART status register to determine the cause of the interrupt.

```
        LDI   #FF     ;Setup RE to point to memory-mapped
        PHI   RE      ;I/O page,
        LDI   #00               and serial I/O board.
        PLO   RE
LOOP1:  LDN   RE      ;read the board repeatedly.
        BR    LOOP1
```

LISTING 1 - Address Decoding and Read Check

```
        LDI   #FF     ;Set up RE to point to memory-mapped
        PHI   RE      ;  I/O Page,
        LDI   #00               and serial I/O board.
        PLO   RE
        LDI   #00     ; Load a data byte into D, and
LOOP2:  STR   RE      ; write it to the board repeatedly.
        BR    LOOP2
```

LISTING 2 - Address Decoding and Write Check

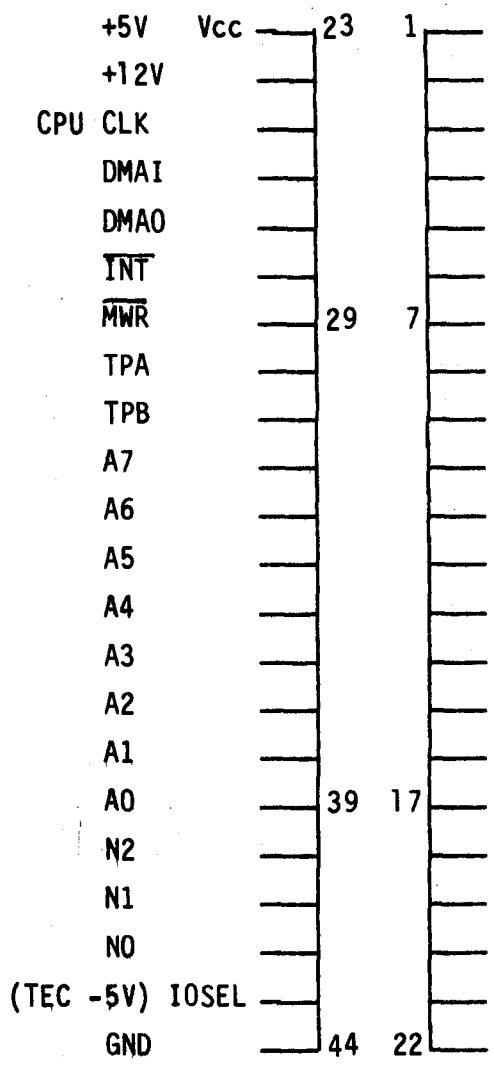
```
BEGIN3: LDI   #01     ;Point to US/ART#0 control/status
                       register
        PL    RE
LOOP3:  LDN   RE      ;read the status.
        ANA   #04     ;isolate TBMT bit.  Empty?
        BZ    LOOP3   ;No. try again.
        LDI   #332    ;Yes. Load a "2" in ASCII,
        DEC   RE      ;Point to US/ART data register,
        STR   RE      ;and load the "2".
        BR    BEGIN3  ;Go do it all again.
```

LISTING 3 - Serial Output Test

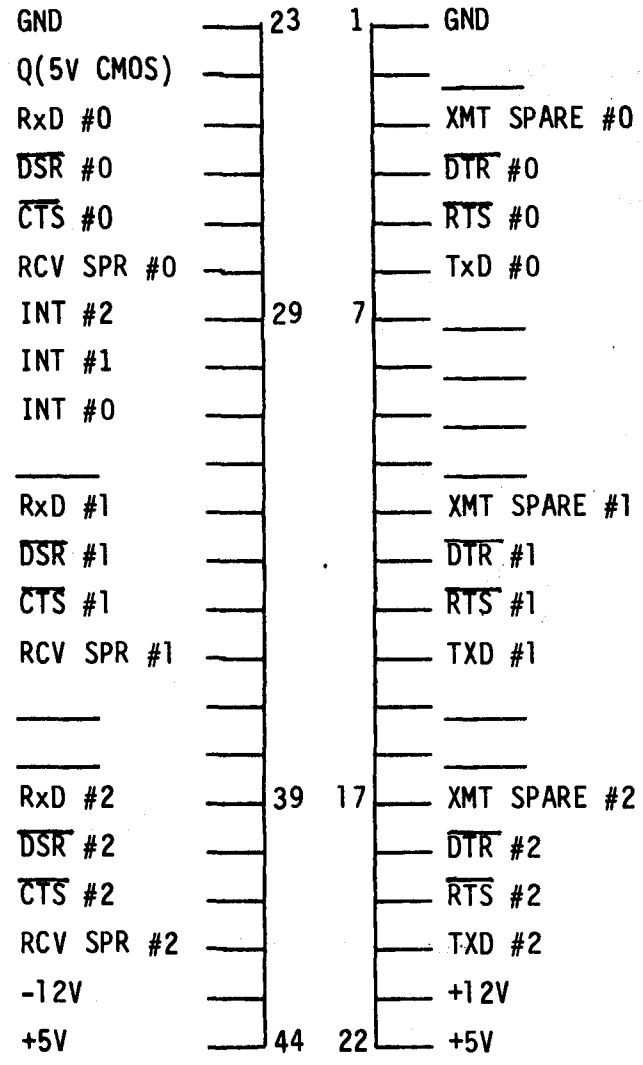
```
BEGIN4: LDI   #81     ;Point to US/ART#0 control/status
                       register.
        PLO   RE
LOOP4:  LDN   RE      ;Read the status
        ANA   #02     ;Have we received a character?
        BZ    LOOP4   ;No.  Go try again.
        DEC   R3      ;Yes. Point to data register.
        LDN   RE      ;Read the received character,
        STR   RE      ;Echo it,
        BR    BEGIN4  ;And go wait for next received character
```

LISTING 4 - Software Echo Test

1802 EDGE CONNECTOR



SERIAL I/O CONNECTOR



MEMORY MAP

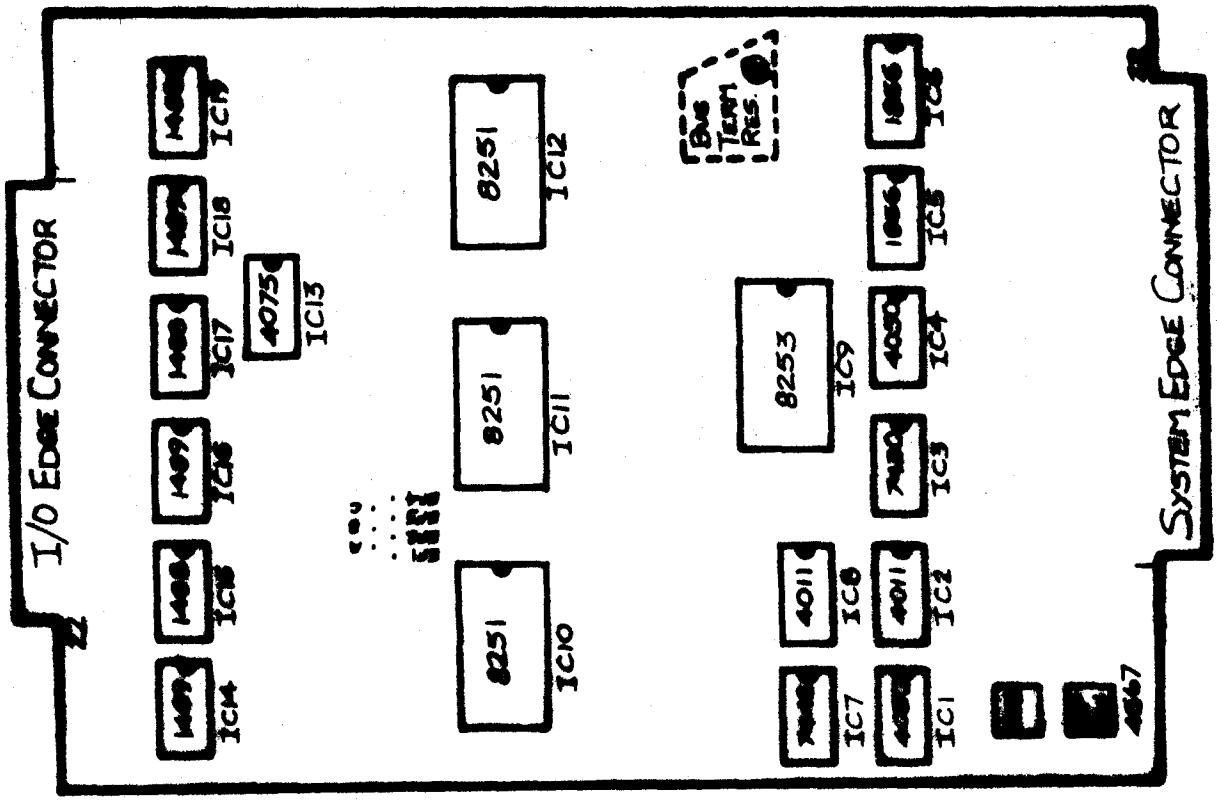
Address	Assignment	Address	Assignment
X0	8251-0 Data Register	X8	CTC - Counter #0
X1	8251-0 Control Register	X9	CTC - Counter #1
X2	8251-1 Data Register	XA	CTC - Counter #2
X3	8251-1 Control Register	XB	Control Point
X4	8251-2 Data Register	XC	N.A.
X5	8251-2 Control Register	XD	N.A.
X6	N.A.	XE	N.A.
X7	N.A.	XF	N.A.

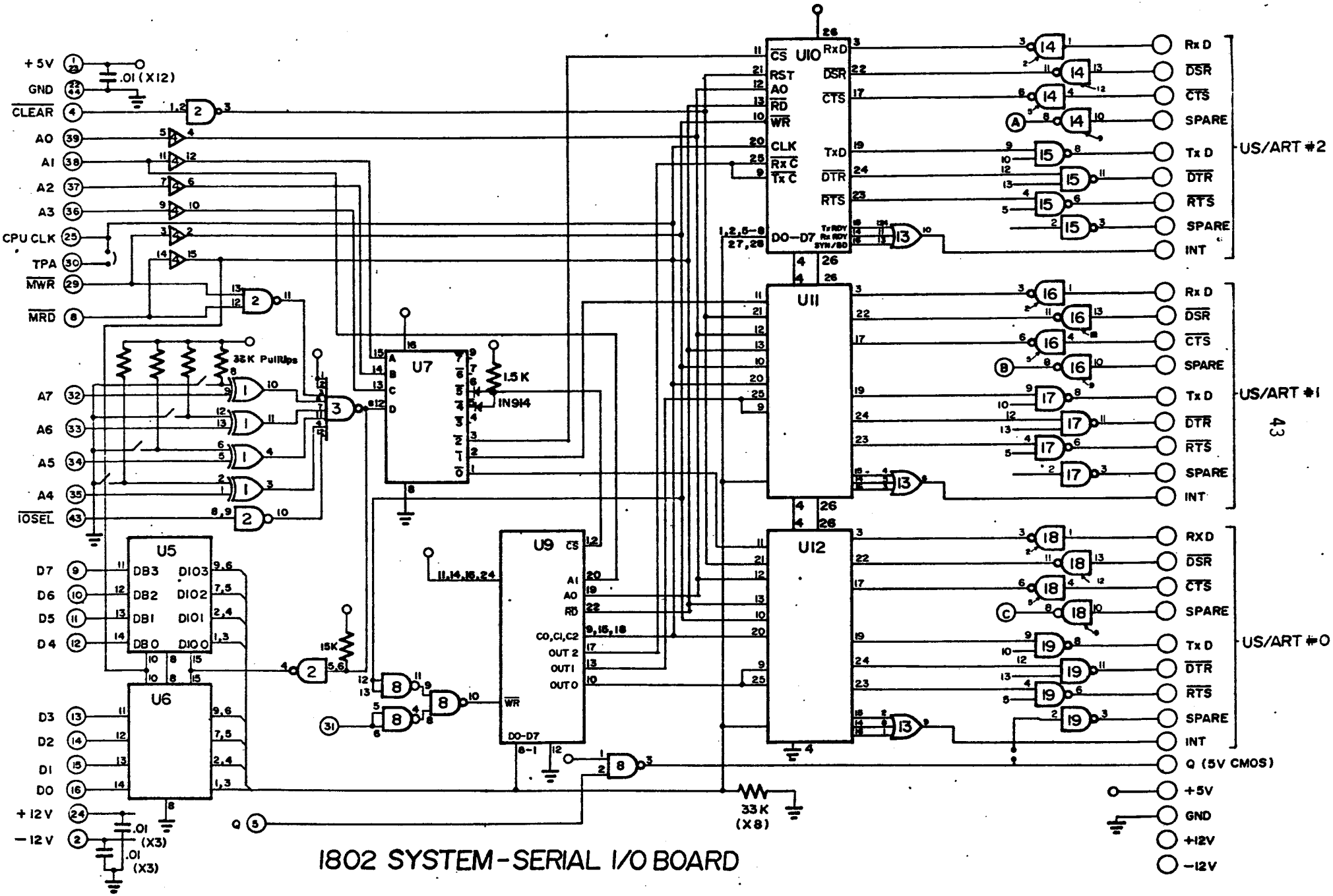
PCB LAYOUT - PARTS LIST AND POWER REQUIREMNTS

POWER CONSUMPTION ANALYSIS

I.C. List				+5V TYP	+5V MAX	+12V TYP	+12V MAX	-12V TYP	-12V MAX
1	4030	1 of	8253	130	130	-	-	-	-
2, 8	4011	3	8251	300	300	-	-	-	-
3	7430	3	1488	-	-	57	75	54	69
4	4050	3	1489	60	78	-	-	-	-
5, 6	1856	1	4075	-	-	-	-	-	-
7	7442	1	7442	28	56	-	-	-	-
9	8253	1	7430	3	6	-	-	-	-
10, 11, 12	8251	2	1856	2	-	-	-	-	-
13	4075	2	4050	-	-	-	-	-	-
14, 16, 18	1489	2	4011	-	-	-	-	-	-
15, 17, 19	1488	1	4030	-	-	-	-	-	-
		TOTAL		523	570	57	75	54	69

		Typical	Max
Power Requirements:	± 12V	60	75 ma
	+ 5V	523	570 ma





1802 SYSTEM-SERIAL I/O BOARD

ACE PRODUCT CATALOGUE

In order to facilitate hardware expansion, and to provide some commonality to the myriad buss and software assignments adopted by Tektron, Netronics, Quest, RCA and home brewists, ACE adopted a standard buss pin assignment and software configuration. Articles in DeFacto and Ipso Facto conform to these standards unless otherwise referenced.

The club has adopted the following 1802 signal outputs:

- 1.79 Mhz clock rate, derived by dividing a 3.58 Mhz colour burst crystal output by two
- Q and EF4 for serial I/O
- Q and EF2 for cassette I/O
- EF1 and Port 1 for 1861 video I/O
- EF3 for handshake, port status
- Port 4 for hex data imput and display

The 44 pin buss adoped by ACE utilized the following pin assignment:

Buss Pin Assignment

1	-+ 5 v.R.	A	-+ 5 v.R
2	-- 12 v.R.	B	-+ 12 v.R
3	- $\overline{\text{WAIT}}$ (RUN)	C	- CPU Clock
4	- $\overline{\text{CLEAR}}$ (LOAD)	D	- $\overline{\text{DMA IN}}$
5	- Q	E	- $\overline{\text{DMA OUT}}$
6	- SC1	F	- $\overline{\text{INT}}$
7	- SC0	H	- $\overline{\text{MWR}}$
8	- $\overline{\text{MRD}}$	J	- TPA
9	- D7	K	- TPB
10	- D6	L	- A7
11	- D5	M	- A6
12	- D4	N	- A5
13	- D3	P	- A4
14	- D2	R	- A3
15	- D1	S	- A2
16	- D0	T	- A1
17	- $\overline{\text{EF3}}$	U	- A0
18	- $\overline{\text{EF2}}$	V	- N2
19	- $\overline{\text{EF1}}$	W	- N1
20	- $\overline{\text{EF4}}$	X	- N0
21	- I/O select	Y	- -5 v.R.
22	- Gnd.	Z	- Gnd.

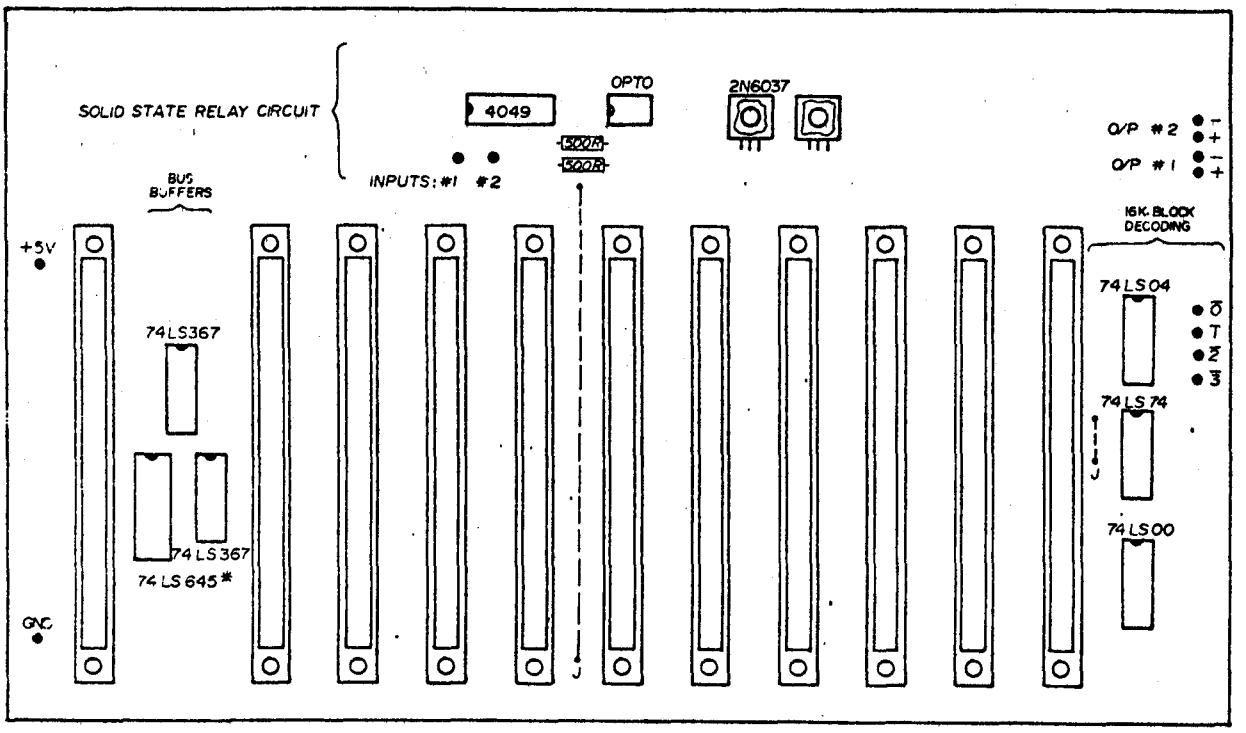
ACE 44 PIN BACKPLANE

Size - 6.0" x 12.0"

Function - 10 slot 44 pin backplane plus a buffered buss slot for micro or system interconnection. 16k Tec memory decoder, solid state cassette recorder control circuit.

Documentation - assembly instructions, buss pin guide.

Power - ± 5 v., ± 12 v., Gnd.



A-C-E STANDARD MOTHER BOARD

ACE 2708 EPROM BOARD

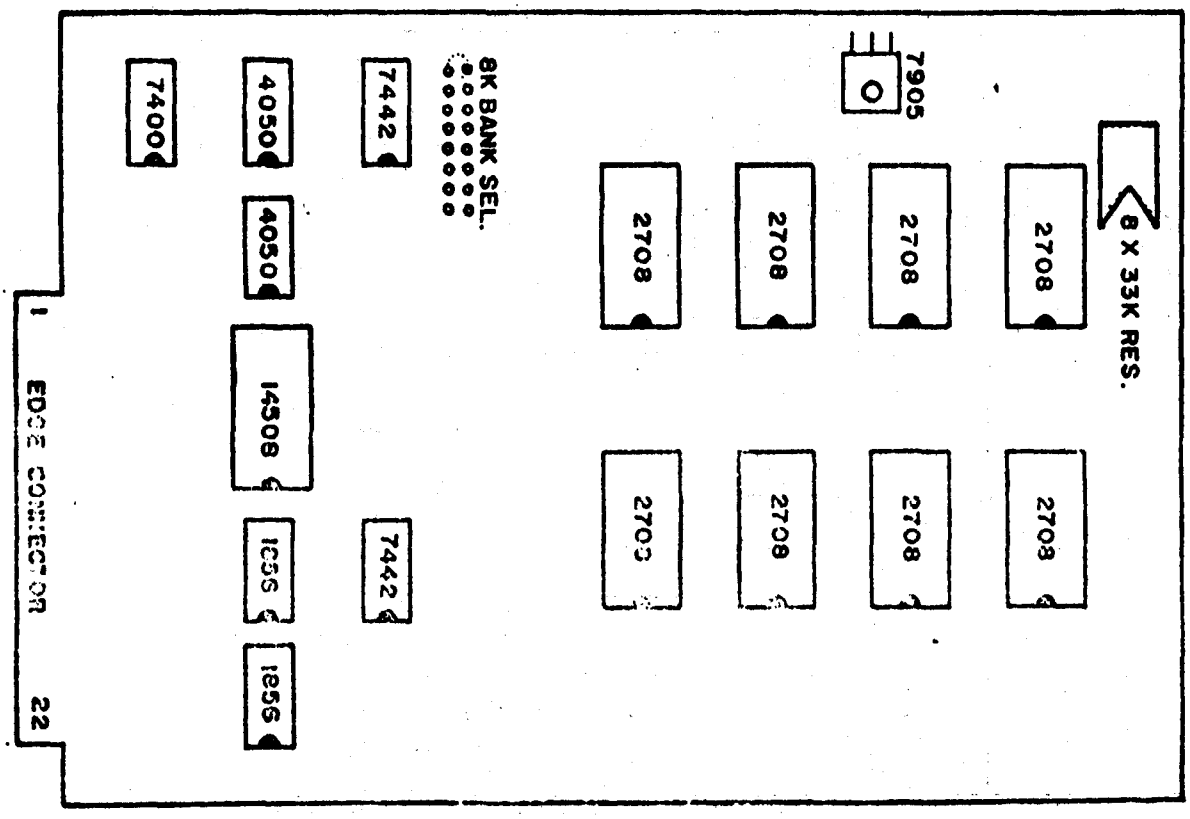
Size - 6.0" x 9.5"

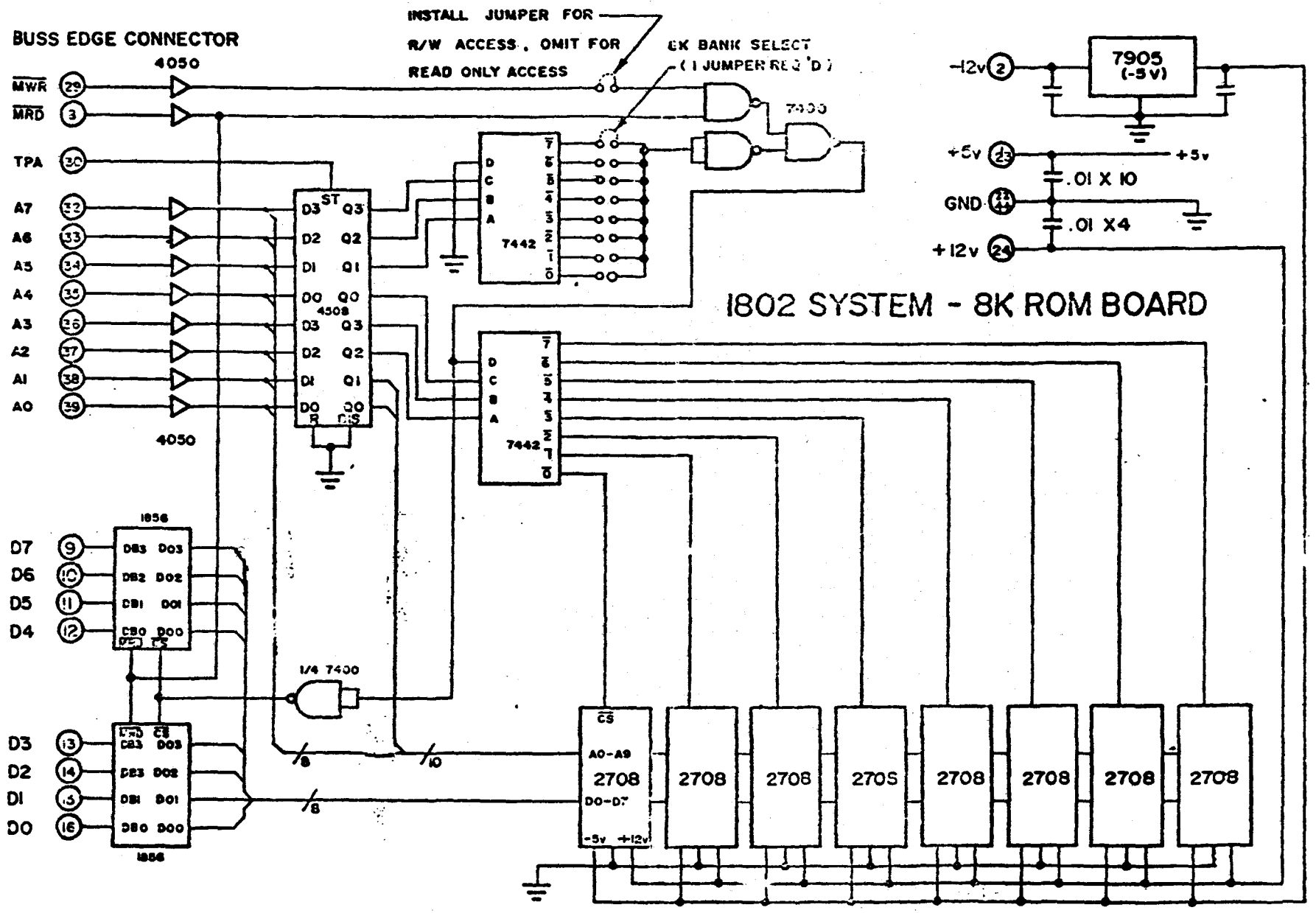
Function - 8k Eprom board, read only. Decoded as one continuous 16k block. Capable of using 1k static RAMS.

Power - ± 5 v.,+ 12v., Gnd.

Documentation - assembly instructions.

Note - with extensive modification, board will accomodate 4 - 2k 2716 Eproms, 8 - 1k 2758 Eproms, or 4 - 2k tri volt Eproms.




BUSS EDGE CONNECTOR
4050
INSTALL JUMPER FOR
R/W ACCESS . OMIT FOR
READ ONLY ACCESS
8K BANK SELECT
(1 JUMPER REQ'D)
MWR 29
MRD 3
TPA 30
A7 32
A6 33
A5 34
A4 35
A3 36
A2 37
A1 38
A0 39
4050
7400
7442
4508
7905
(-5V)
-12v 2
+5v 25
GND 12 13
+12v 24
.01 X 10
.01 X4
+5v
1802 SYSTEM - 8K ROM BOARD
1856
D7 9
D6 10
D5 11
D4 12
D3 13
D2 14
D1 15
D0 16
1/4 7400
2708
A0-A9
D0-D7
-5v +12v
8
10

ACE VDU/MEMORY BOARD - Ver. 2

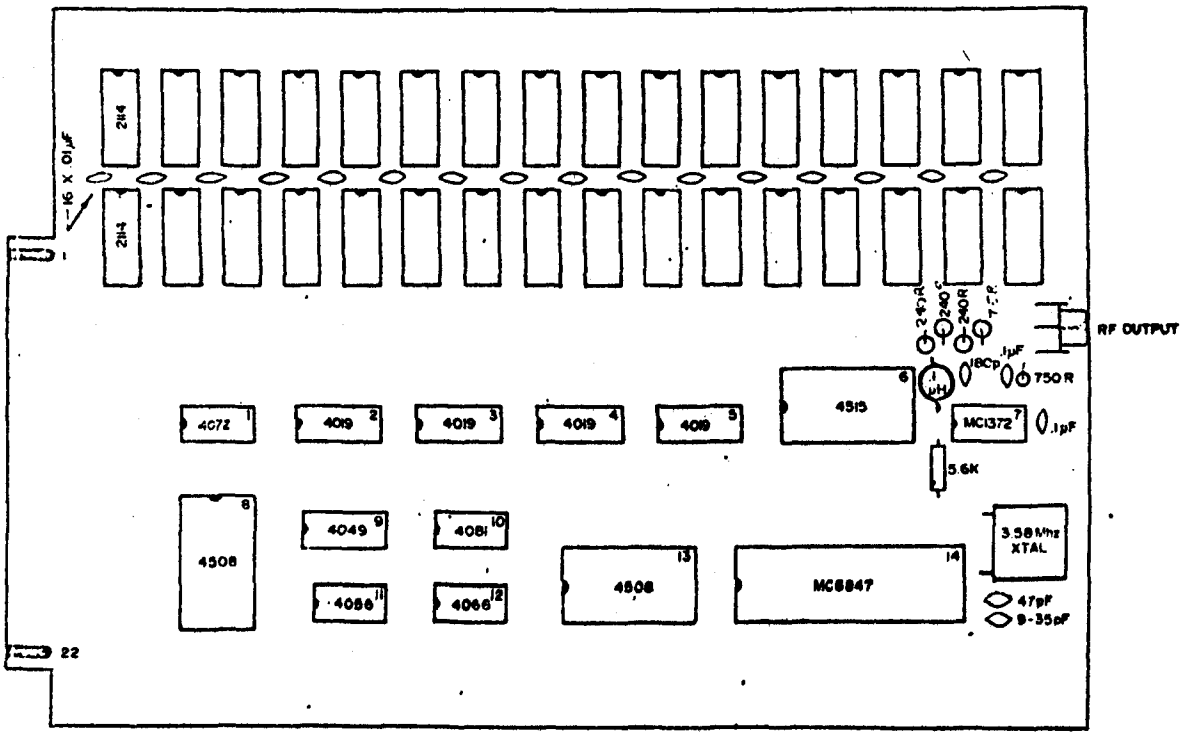
Size - 6.0" x 9.5"

Function - EITHER 16k static memory board, 2114 Rams, addressed as a continuous 16k block

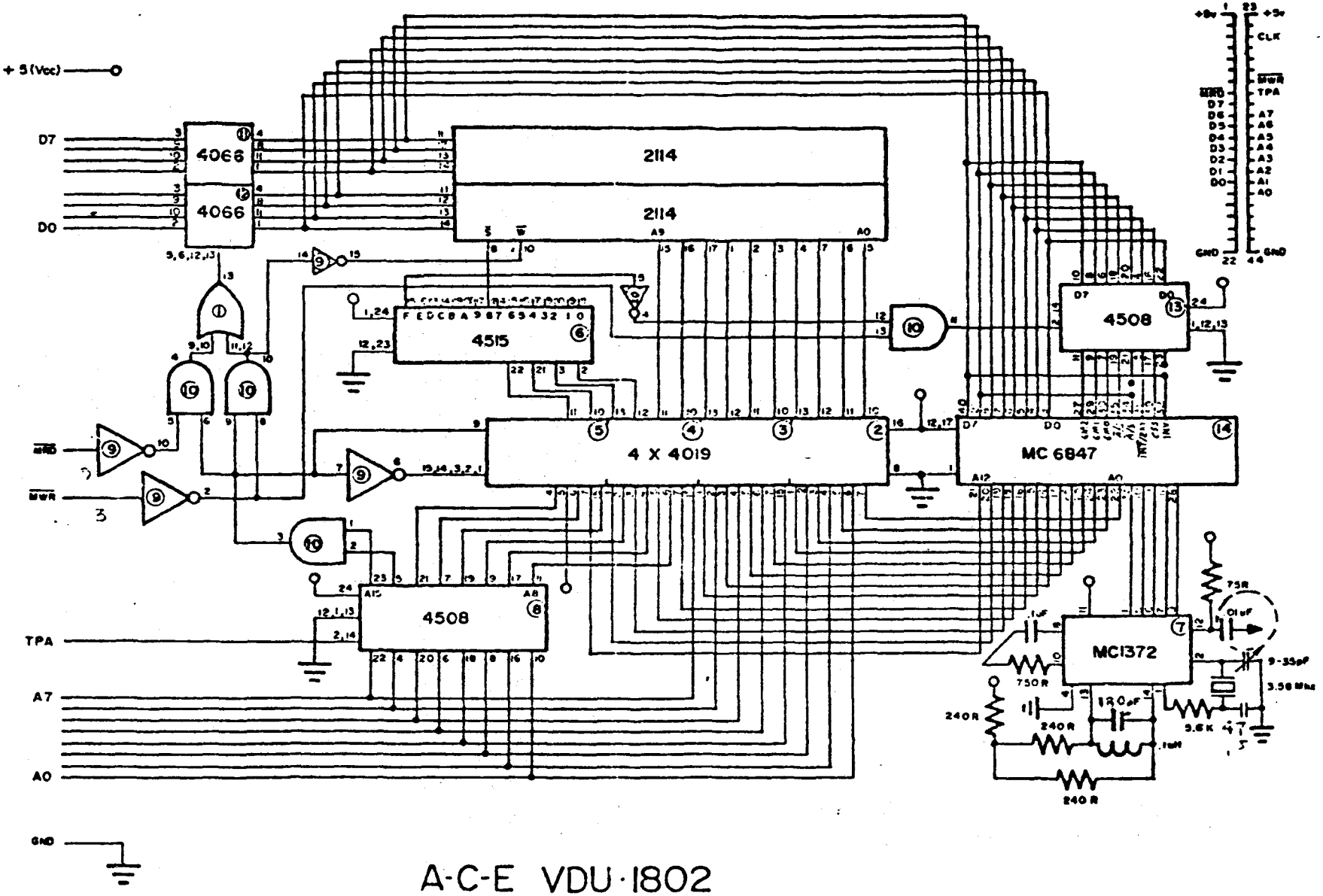
OR colour video display unit, 6847VDU, with 1 to 6k video buffer addressed from E000. Video RF modulator provided on board.

Power - +5 v, Gnd.

Documentation - assembly instructions, trouble shooting procedure, MC 6847 documentation, memory test program.



A-C-E VDU 1802 VERSION 2


+ 5 (Vcc)
D7
D0
4066
4066
2114
2114
4515
4508
4 X 4019
MC 6847
4508
MC1372
TPA
A7
A0
GND
75R
240R
240R
240R
9.6K
3.58 MHz
9-35pF

A-C-E VDU·1802

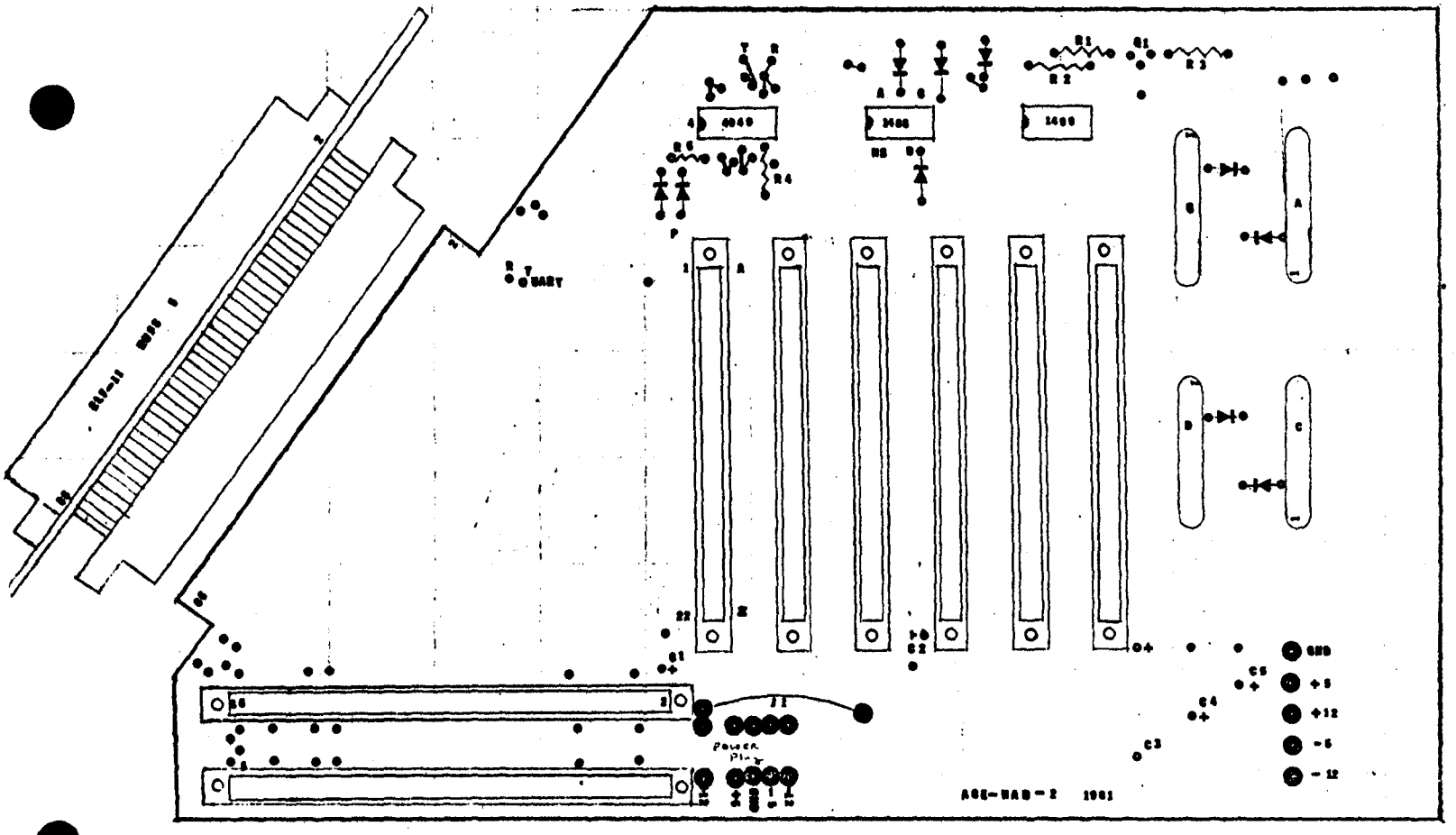
ACE NETRONICS - ACE - ADAPTER BOARD

Size - 7.0" x 12.0"

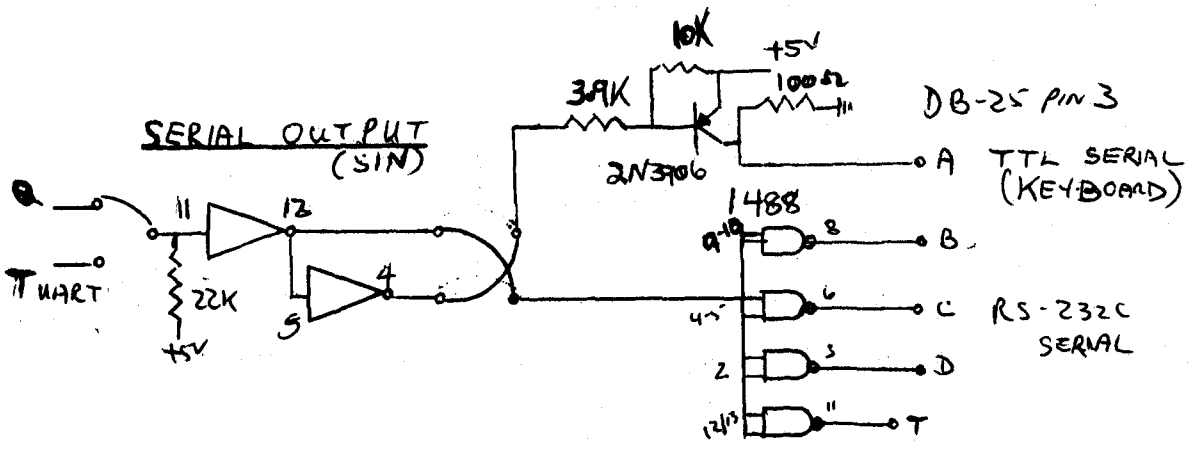
Funtion -a hard wire interface betwwen the 86 pin Netronics buss and the ACE 44 pin buss, and a TTL or RS-232C serial driver, receiver, and handshake circuit. Buss power distribution also available.

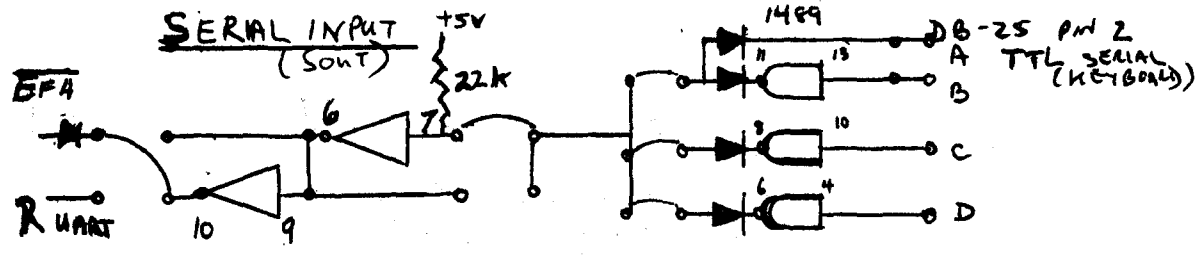
Power - ± 5 v., ± 12 v., Gnd.

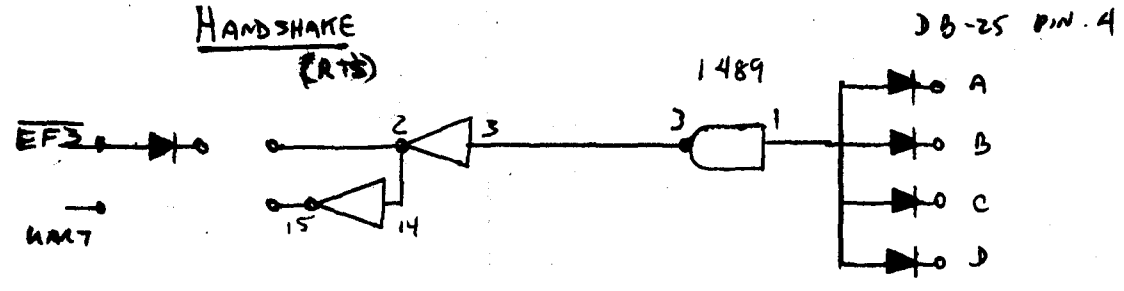
Documentation - assembly guide, ELF 11 modification instructions.



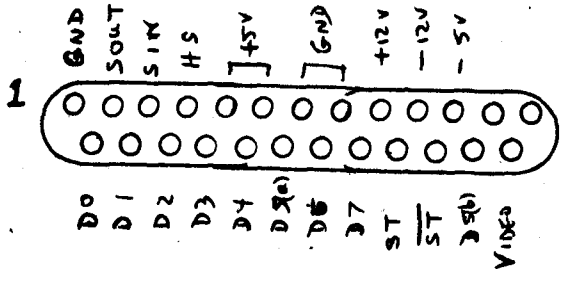
NETRONICS - ACC ADAPTER BOARD VER 2. 1981.



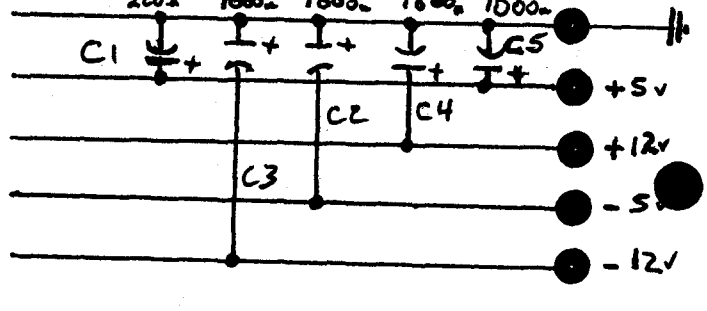




DB25 CONNECTOR



Power Supply



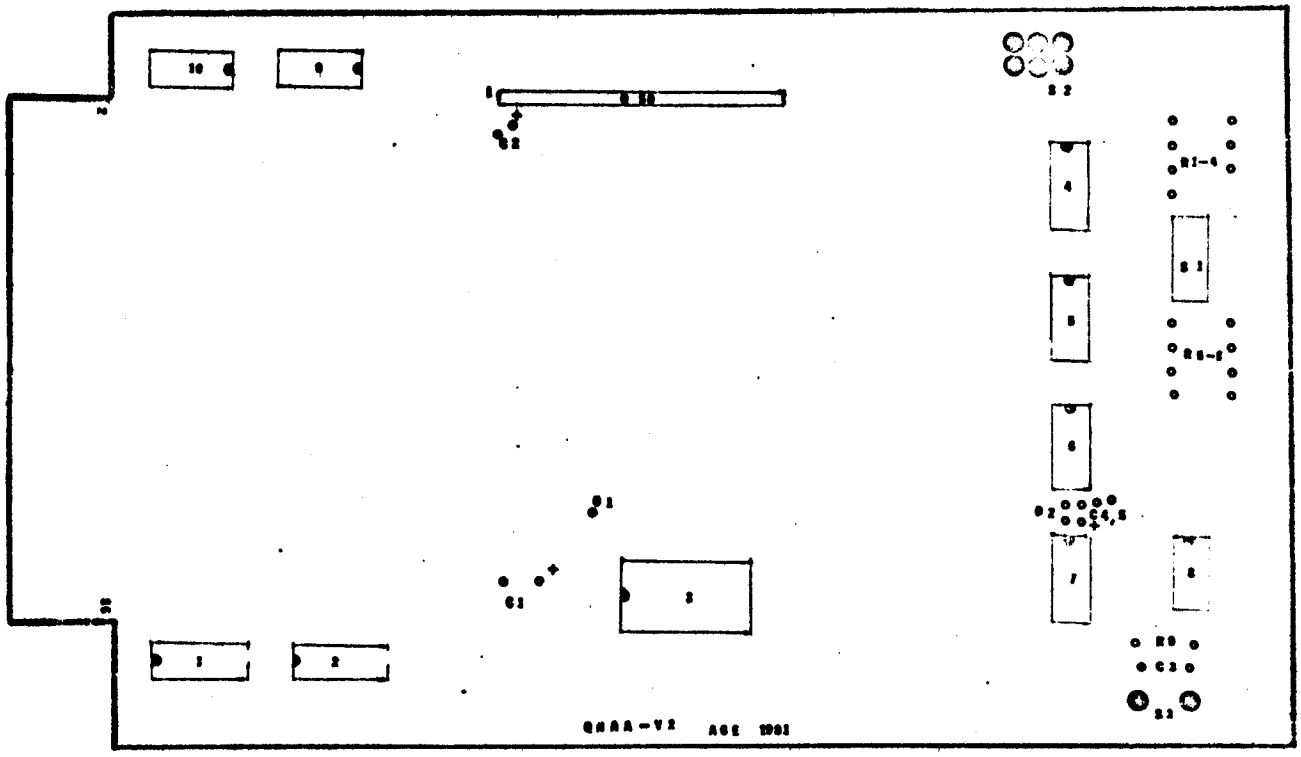
ACE QUEST - NETRONICS - ACE ADAPTER BOARD

Size - 5.0" x 11.0"

Function - a hard wire interface between the NAB 86 pin buss and the Quest 50 pin buss of the Super Expansion Board. In addition, 4k RAM and 4k EPROM (2716) switchable decoding, plus a separate fixed 2k EPROM monitor and 1k RAM stack and 1k Memory Mapped decoding are provided.

Power - ± 5 v., ± 12v., Gnd.

Documentation - Assembly instructions, Quest Super Expansion Board modification instructions.



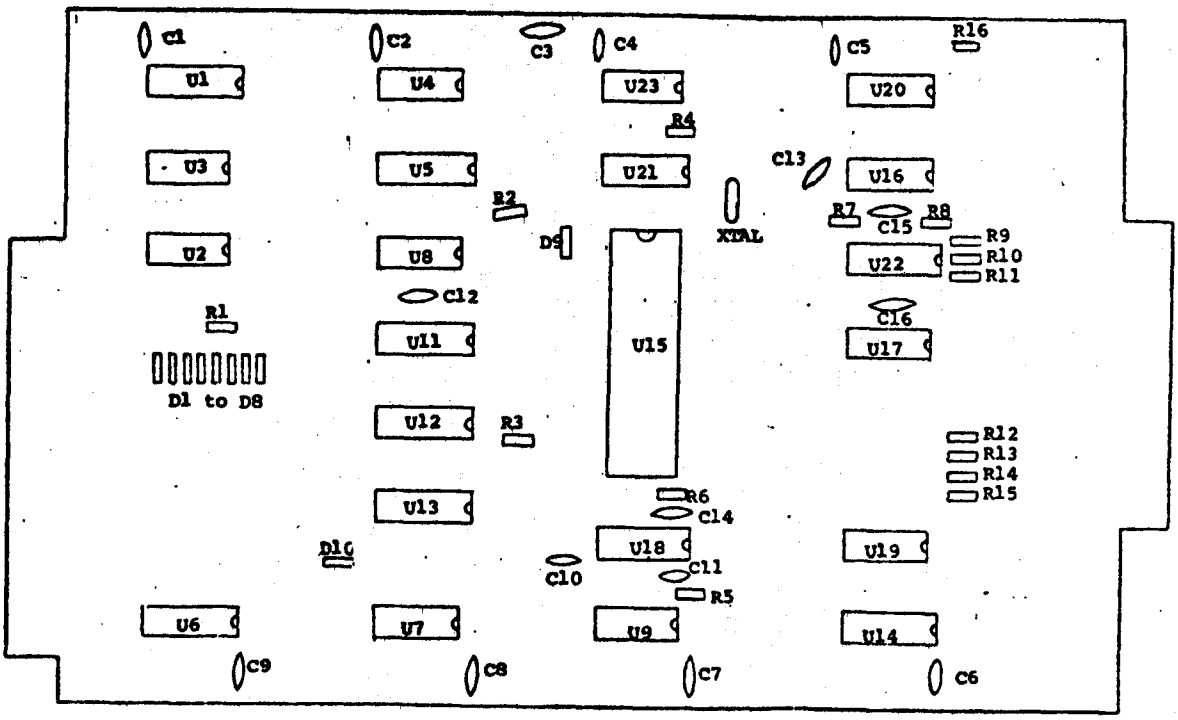
ACE DISK CONTROLLER BOARD

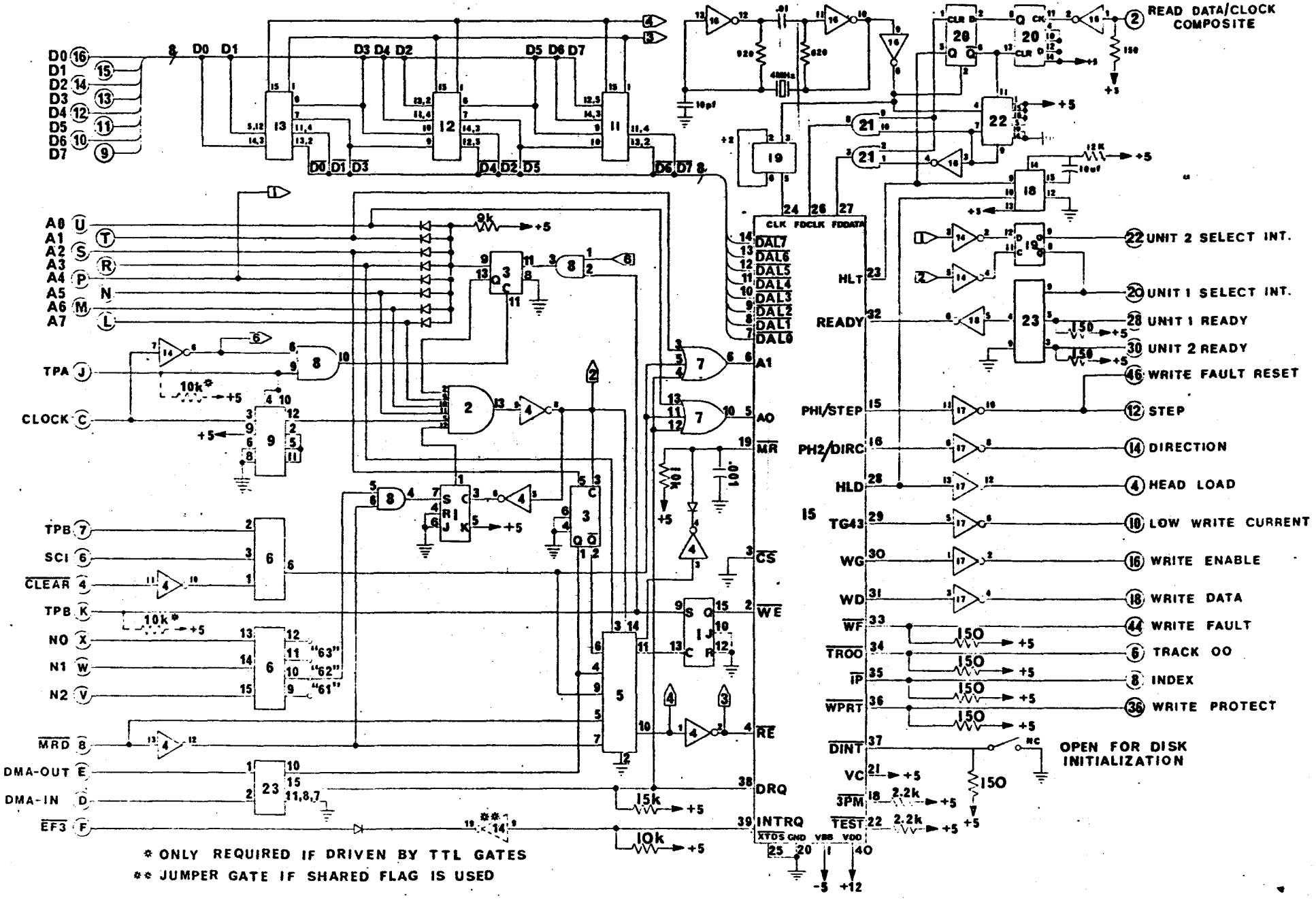
Size - 6.0" x 10.0"

Function - DMA oriented 8" Disk controller for the 1802.
Singe sided, single density WD 1771 Controller chip.
Designed to support two 8" Disks, jumperable Disk
Interface will accommodate any 8" Disk. Probably
could be modified to support 5.25" Disks.
ELF 11 users require DMA Adapter board for buss interface.

Documentation - assembly instructions, mini DOS, DOS exerciser
program.

Power - ± 5 v., + 12v., Gnd.




D0 D1 D2 D3 D4 D5 D6 D7
A0 A1 A2 A3 A4 A5 A6 A7
TPA
CLOCK
TPB
SCI
CLEAR
TPB
N0
N1
N2
MRD
DMA-OUT
DMA-IN
EF3
READ DATA/CLOCK COMPOSITE
UNIT 2 SELECT INT.
UNIT 1 SELECT INT.
UNIT 1 READY
UNIT 2 READY
WRITE FAULT RESET
STEP
DIRECTION
HEAD LOAD
LOW WRITE CURRENT
WRITE ENABLE
WRITE DATA
WRITE FAULT
TRACK 00
INDEX
WRITE PROTECT
OPEN FOR DISK INITIALIZATION
DAL7
DAL6
DAL5
DAL4
DAL3
DAL2
DAL1
DAL0
HLT
READY
PHI/STEP
PH2/DIRC
HLD
TG43
WG
WD
WF
TROO
IP
WPRT
DINT
VC
3PM
TEST
A1
A0
MR
CS
WE
RE
DRQ
INTRQ
CLK FDCLK FDDATA
XTDS GND VBB VDD
"63"
"62"
"61"
* ONLY REQUIRED IF DRIVEN BY TTL GATES
** JUMPER GATE IF SHARED FLAG IS USED

ACE DMA ADAPTER BOARD

Size - 2" x 3.5"

Function - provides DMA flip flop circuit to 1802 from buss as well as from 1861 video chip and HEX PAD.
Allows ELF 11 users to use buss oriented DMA devices, such as ACE DISK CONTROLLER BOARD.

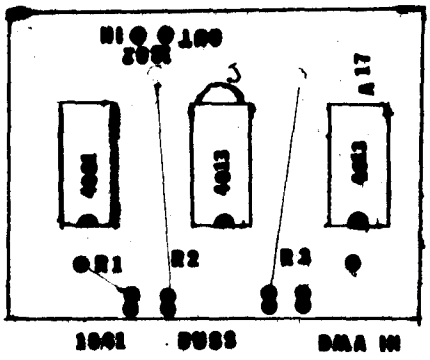
Documentation - assembly instructions.

Power - + 5v., Gnd.

ELF II DMA ADAPTER BOARD.

Sept. 1981 - MEF/ACE.

Parts layout



Parts List

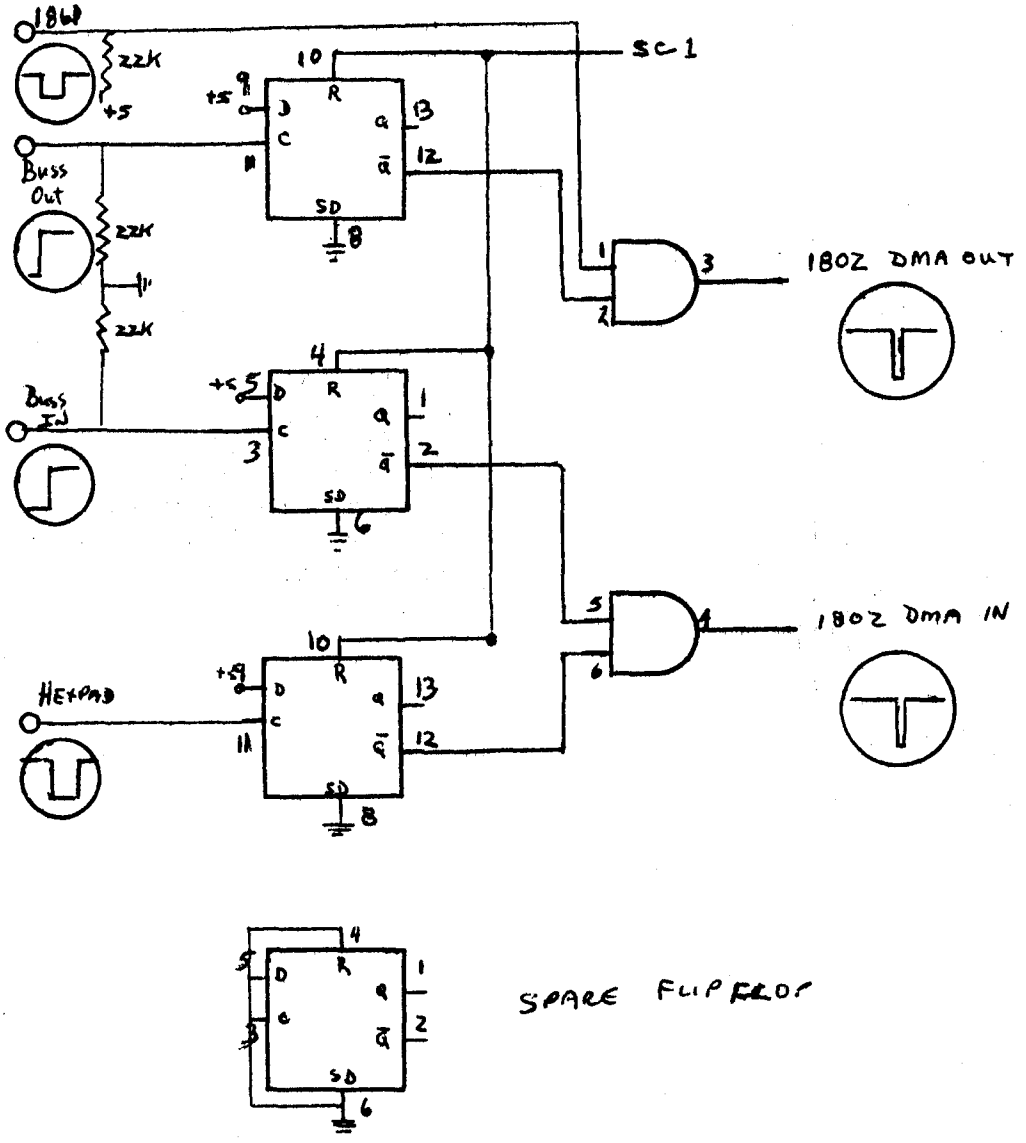
1-4081 AND gate
2-4013 D Flip Flop
2- 14 pin dip sockets
1- 14 pin wire wrap socket
3- 22k ¼w. resistors.
connecting wire.

SCHEMATIC

LOGIC TABLE - AND

in 1	in 2	out
0	0	0
0	1	0
1	0	0
1	1	1

- all inputs normally high
-a negative signal on any input will drive DMA pin low signalling a DMA operation.



SPARE FLIP FLOP