IPSO FACTO

ISSUE #4
JANUARY, 1978

TABLE OF CONTENTS

Editor : Tom Crawford
Invaluable Assistants : Wayne Bowdish and all contributors to this issue.

All Newsletter correspondence should be sent to:
Tom Crawford
50 Brentwood Drive
Stoney Creek, Ontario
CANADA L8G 2W8

We wish to express our thanks to Xerox Corporation and to Dominion Foundries and Steel Co. Ltd. for their assistance in the printing of this issue of IPSO FACTO.

EDITOR'S REMARKS

FEB. 4, 1978

We have received many comments in the past month concerning Issues 2 and 3 of IPSO FACTO. I am pleased to say that almost all of the reader feedback has been favourable (1 person, though, thinks our Newsletter is too big!)

The last club meeting, held on January 10th, saw several items discussed with great interest. Members are getting tired of watching their console lights flash, and entering machine language programs 1/2 byte at a time on a hex keypad. They are starting to think seriously about running high-level languages, and the hardware system nescessary to support them.

The prime consideration now is to pick up an alpha-numeric keyboard and display. To this end, several members will be investigating the availability and price of surplus and used video generators, Teletypes (Baudot and Ascii), and keyboards. We hope to hear from them at the next meeting on February 9th.

We will also be having a flea market at the February 9th meeting, and if it proves successful, we will probably set it up as a monthly affair. So bring any items you are tired of, and some one will probably buy them from you. There is no dealer's fee charged by the club.

We have scheduled club meetings and tutorials for the next 4 months. You will find a copy of the schedule at the end of the Minutes of the last club meeting, in this issue of IPSO FACTO. If you aren't sure where the Stelco auditorium is, call one of the Executive for directions.

One article that everyone should read can be found on pgs. 30-33 of the October 1977 issue (#19) of DR. DOBB'S JOURNAL, or on pgs. 65-69 in the February 1978 issue of POPULAR ELECTRONICS. The article is by Edward M. McCormick, and deals with several items of simple hardware and software for the 1802.

In early January I received a copy of Ross Wirth's Booklet of Software for the 1802, in which IPSO FACTO and several club members are listed as sources of significant software. In addition Ross will be passing on any further software he receives to us in the form of articles for IPSO FACTO.

For those who cnnot attend club meetings in Hamilton, but would like to get in touch with club members living in their area, I suggest you write me a "Letter of Contact" for inclusion in the Letters to the Editor section of IPSO FACTO. Simply state the geographic area you live in, and include your name, address and phone number. For example, you might live in Ottawa, and would be interested in getting together with any members who live within your local telephone calling area.

Incidently, I would like to point out that the "Tektron Newsletter Issue #1" is Issue #1 of IPSO FACTO before it was called IPSO FACTO. The name "Tektron Newsletter" is no longer in use.

One very interesting source of software for the 1802 can be found in RCA's User's Manual for their VIP home micro-computer. This is a computer based on the 1802 chip combined with an 1861 video display chip. The software I have seen shows an Operating System for hex keypad in 512 bytes of memory, and a 512 byte Interpreter program to allow use of RCA's "CHIP-8" high-level language. There are also 20 sample game programs to get you started in CHIP-8. A very interesting package for those of you with an 1861 chip. Check with your local RCA dealer.

A TEXT EDITOR

Wayne Bowdish

The editor of IPSO FACTO has asked me to write an article on a text editor which I have been working on. This program would allow you to create ASCII text files for input to a resident assembler or compiler or data files. I think that about 2K bytes of memory would be required for the editor.With the large memory board (soon to come I hear) and a good cassette interface this program would provide a very useful function.

I have decided to write the article in the form of a users manual (although somewhat shortened for publication purposes). If there is any interest I will write some future articles describing some ideas on implementing the editor.

FILES

EDITR views a file as a series of lines of ASCII characters . A line is terminated by a carriage return and line feed. Each line begins with a 1 to 5 digit line number in the range of 1 to 32767 (32k). Note that a line number may have leading zeros which are ignored by EDITR. A line may consist of up to 80 ASCII characters (including the line number but not the carriage return line feed sequence, ie. the line could actually be up to 82 ASCII characters in length).

COMMANDS

The EDITR commands are described in this section. They will be listed in alphabetic order. Certain conventions have be adopted in this section.

<return>	represents pressing the carriage return key . This key is usually labled RETURN , RTN or CR .
k , m , n	The lower case letters "k", "m", and "n" are used to represent line numbers in the range 1 to 32767 (32k).
inc	This represents a line number increment.Increments may be in the range 1 to 32767 (32k).
<text>	represents a string of ASCII characters which the user types in.The string may not contain line feed or carrige return characters since these are used as end of line indicators.
_	The underscore character will be used to indicate a space character where spaces are required.Spaces are used to separate commands from their arguments and to separate multiple arguments.

NOTE in all cases if line number k and line number m are specified line number k must be less than or equal to line number m .

C - Copy specified lines into new location

The general form of the copy command is:

C_k_m_n_inc <return>

This command copies lines k through m to a new location starting at line number n and incrementing the new line numbers by inc. If inc is not present the standard default of 10 is assumed. Lines k through m are not altered. There must not be any lines with line numbers between n and the largest line number created by the copy operation.

ex. if the user wanted a copy of lines 3, 4, and 5 at line number 100, 110, and 120 the command
C 3 5 100 10
would be used. The file now contains lines 3,4,5,100, 110, and 120. Note that in this example the inc value of 10 was not required since the default value for an increment is 10. ie. the command could have been
C 3 5 100

D - Delete lines

The general form of the delete command is:

D_k_m <return>

This command deletes all lines from line number k through line number m inclusive. If m is not specified then only line k is deleted. Lines which are deleted are effectively removed from the file and may not be recovered.

ex. D 3 9 delete lines 3 through 9 inclusive
D 77 delete line 77

G - Get file from cassette

The general form of the get command is:

G <return>

This command reads the next file from the cassette unit into the program buffer for editing. Further definition of the format of tape data will be required before the format of this command can be completely defined.

I - Insert lines

The general form of this command is:

I_k_<text> <return>

This command inserts the specified line <text> into the file at line number k . If line k exists when an insertion is attempted an error message is typed.

ex. I 55 THIS IS LINE 55 line number 55 is added to the file

M - Move lines to new location

the general form of the move command is:

M_k_m_n_inc <return>

This command move lines k through m to a new location in the file starting at line number n and incremented by inc. The original lines k through m are then deleted from the file. If inc is ommited the the standard default increment of 10 is used. In order to move only 1 line k and m should be set equal.

N - reNumber lines

The general form of the renumber command is:

N_inc <return>

This command will renumber all of the lines currently in the file. The first line will have line number inc and the successive line numbers will be incremented by inc. If inc is not specified the the first line will be line number 10 and the default increment of 10 will be used.

ex. If the user has a file containing lines 97 , 105 , and 213 the command:

N 100

would renumber the lines to 100 , 200 , and 300. The command

N

would renumber the lines to 10 , 20 , and 30 .

P - Put file on cassette

The general form of the put command is:

P <return>

This command will write the file currently in the editers buffer onto the cassette unit. Further definition of the format of data on the tape will be required before the P command can be further defined.

R - Replace an existing line

The general form of this command is:

R k <text> <return>

This command is used to replace an existing line in the file.This command may be thought of as performing a Delete command (D k) followed immediatly by an Insert command (I k <text>).

```
ex.  R 34 NEW LINE 34          !delete line 34 if it was
                               !in the file and insert
                               !the new text
```

S - Scratch current file

The general form of this command is:

S <return>

This command deletes all lines in the editors buffer and initializes the editor. This command would be used if the user was editing several files in 1 editing session. After each file had been ed ited and saved on the cassette tape an S command would be issued to initialize the editor for the next file.

T - Type lines on output device

The general form of the type command is:

T_k_m <return>

This command types the specified lines of the file on the users output device. If both k and m are given then lines 1 through m inclusive are typed.If m is not specified then only 1 line is typed. If no arguments are entered then the whole file is typed.

ex.	T 50 100	type lines 50 through 100 inclusive
	T 63	type only line 63
	T	type the entire file

EXAMPLES OF COMMANDS

The following section uses examples to demonstrate the use of the various editor commands. User input will be underlined while any editor output will not be. Comments will be made on the right side of the page where necessary. These comments will be in lower case and will be preceeded by an exclamation mark ! . It should be noted that each line that the user types must be terminated with a carriage return.In all of the examples it is assumed that the user has previously loaded the editor and has started it.

EXAMPLE # 1: Creating a new file

The user wishes to create a file and then later place this file on a cassette tape. Note that the editor outputs a prompt character when it is ready for input (the "@" character).

```
@I 10 THIS IS THE                    !the user begins inserting
@I 20 FIRST EXAMPLE OF               !text into the editors
@I 30 THE USE OF EDITR               !buffer
@I 40 TO CREATE A FILE
@D 10 20                             !a change is desired so 2
                                     !lines are deleted
@I 1 EXAMPLE # 1                     !a new line is inserted
@T                                   !the file is now typed to
00001 EXAMPLE # 1                    !check the changes
00030 THE USE OF EDITR
00040 TO CREATE A FILE
@N                                   !resequence the file
@T                                   !and retype the whole file
00010 EXAMPLE # 1
00020 THE USE OF EDITR
00030 TO CREATE A FILE
@P                                   !output file to cassette
@
```

The editing session is now complete with the new file stored on the cassete.

EXAMPLE # 2 - Editing an existing file

This example is an editing session on an existing file. The user has positioned the cassette tape so that the next file read is the one which is to be edited.

```
@G                                  !read file off of cassette
@I                                  !type the whole file
00010 EXAMPLE # 1
00020 THE USE OF EDITR
00030 TO CREATE A FILE
@R 30 TO MODIFY A FILE              !a line is changed
@I                                  !retype edited file
00010 EXAMPLE # 1
00020 THE USE OF EDITR
00030 TO MODIFY A FILE
@P                                  !replace file on cassette
@
```

This example demonstrated a method for editing an existing file. Note that before the P (put) command the tape should have been re-wound otherwise the new file would have been written onto the tape immediatelly after the first. Multiple copies of a file is usually a good idea since tape data can deteriorate and valuable data will be lost.

ERROR MESSAGES

?CHAR?	illegal character in command string
?LINE?	line number out of range or non-existant
?CMD?	illegal command character
?TAPE?	cassette tape error detected
?RENUM?	file requires renumbering before insert can can be done
?FULL?	editor buffer full

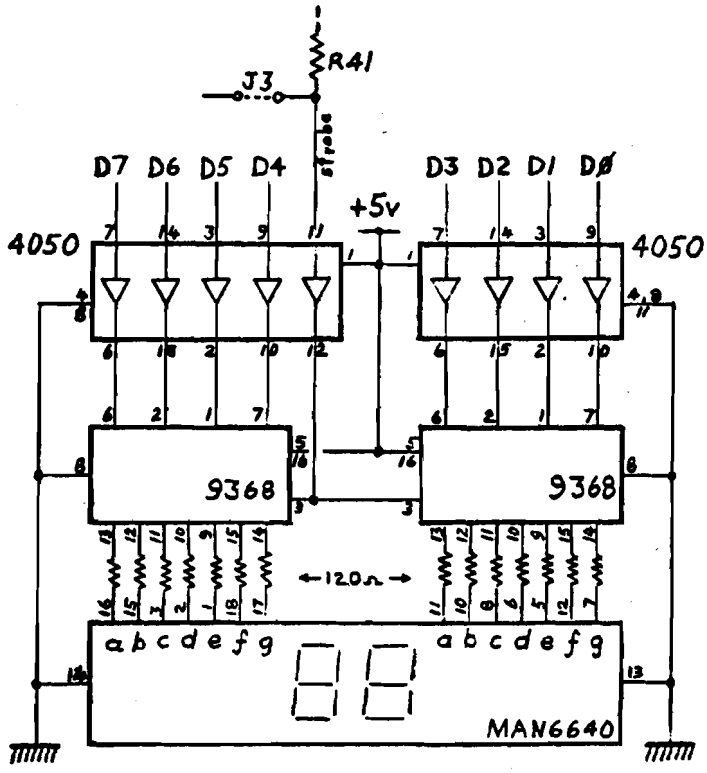
ECONOMICAL HEX-DISPLAY

Cec Williams

The 1978 Electronic Experimenter's Handbook shows another version of the original COSMAC ELF. It uses a simpler and more economical data display than the original ELF, that specified expensive Hewlett-Packard integrated hex-display chips.

As no advertisement could be found for the dual National hex-display specified, the circuit was adapted to the Monsanto dual hex-display, their MAN6640, a common cathode device. This has 2 orange, 0.56" high digits. The circuit produces the hex numerals and alphas, with 'b' and 'd' being in lower case. The original 8 data LED's should, of course, be removed.

The four IC's and the one dual hex-display are obtainable from James Electronics, California, for just over $10. This gives a one-digit price of about $5, which compares favourably with the H-P price of about $15 - $20 per digit (when obtainable). The circuit is easy to breadboard; it can also be used to replace the address LED's, if desired.



A MONITOR FOR ASCII KEYBOARD AND DISPLAY

Tom Crawford

In this article I shall describe a monitor program I am in the final stages of debugging. First, though, I would like to elaborate on the reasons for having a monitor.

The first contact most people have with their own micro-computer is through a hex keypad and a row of LEDs. This means of I/O works OK as an introduction to the machine, but soon becomes tedious for those who attempt any serious programming. Another irritating factor is the loss of memory contents each time the power is switched off, resulting in the need to reload all the code you are working on, 1/2 byte at a time, through the keypad.

The solution to the 2nd problem for most people requires adding a cassette interface to your microcomputer. This allows speedy reloading of software each time you use your system.

The solution to the problem of easier input and output usually takes the form of a serial ASCII terminal. This can take many shapes, depending upon the size of your wallet. In my case, I have constructed a TVT-2 video terminal (also known as a CT-1024) with a 64 key keyboard, and a serial interface. The serial interface adds some complexity to the hardware, but allows substitution of a hard copy terminal such as a Teletype when my budget permits, simply by changing connectors.

The ASCII terminal is a general purpose terminal, in the sense that it allows the entry of machine language data now, and can be used for higher level languages employing the full ASCII character set at a later date.

The problem at this point is to tie together the ASCII terminal, the cassette interface, and the micro-computer into a system which can stand alone for machine language use, and which will provide the detailed I/O routines which service the terminal and the cassette, for the use of the higher level language programs. This is the purpose served by a monitor program.

Virtually every computer system has a monitor, and many of the simpler ones provide similar features. I chose a monitor I was familiar with, The Motorola MIKBUG monitor, and re-wrote it for an 1802 with my particular I/O arrangement. The result has been nick-named "RIKBUG".

THE RIKBUG MONITOR

RIKBUG, as currently written, is 750 bytes long, and interfaces to a serial ASCII terminal via the Q-line (output) and EF1 (input). The routines assume a frame composed of 1 start bit, 7 data bits, 1 parity bit (even parity) and 1 stop bit. RIKBUG initially determines the baud rate of the serial device by timing the start bit of the first character entered.This feature has been tested at 300 and 1200 baud, and works quite well.

RIKBUG also handles a KC standard cassette interface as written up in Issue #3 of IPSO FACTO. Q is the output, while EF2 is data input and EF4 is clock input. The routines perform a software UART function, which will allow use of the cassette interface at 300, 600, or 1200 baud (irrespective of the baud rate of the ASCII terminal).

RIKBUG provides the following functions, each started by the associated key letter:

G - Go to target program
L - Load from tape
M - Memory change
P - dump to tape
R - display contents of target stack

The monitor is highly subroutinized, and makes use of RCA's SCRT routines (modified to save the D-register contents). Therefore any user-generated software can make use of the terminal and tape I/O routines simply by making a SCRT call, with data passed back and forth in the D-register. (This assumes RIKBUG is co-resident in memory with the user's program, of course !)

RIKBUG is not yet completely debugged, as I mentioned earlier. The G, M, P and R routines all work satisfactorily. The L function reqires mods to make use of the external clock on EF4; I am confident it will be working shortly.

I am willing to make copies of RIKBUG available to anyone seriously interested. I will supply an assembly language listing, including extensive comments and the assembled machine code. No cassette tapes are available yet. Please send a 9" X 12" SASE (50 cents Canadian postage only please) to: TOM CRAWFORD, 50 BRENTWOOD DR., STONEY CREEK, ONT., CANADA L8G 2W8

(The listing is 28 pages long, so if you scrimp on the envelope or the postage, you won't get it).

This software was assembled using Wayne Bowdish's 1802 cross-assembler. Some extended instructions ("+" instructions) are used, such as "+CALL" and "+RETRN" . These are self-explanatory, for the most part. Further details can be obtained by contacting Wayne. See his article in Issue #2 of IPSO FACTO.

LETTERS TO THE EDITOR

(Ed. Note: The following letters have been edited slightly to conserve space. The Editor apologizes for any errors or omissions which result. TC)

Dear Sirs:

I have heard of your club and newsletter in the last issue of Popular Electronics. I own an 1802 based ELF II and am very interested in your efforts.

There is a local hobbyist computer club trying to get a start here. I will relay any info to the club.

I have several projects under way currently and would be interested in contributing articles to your letter. (please do.TC)

Thank you very much., Claud Hesselman, 805 "D" St.
Chesapeake, Virginia, U.S.A. 23324

Dear Tom:

I read your announcement in the Nov. issue of Byte. I wish to obtain more information on your 1802 Users' Group. I am in the process of building a computer based club using an 1802 computer we built from scratch. As you can see any info regarding your users' group would be appreciated.

Thank you, Mark Moore, 36 Waverley Dr., Guelph, Ontario, N1E 6C8

Sirs:

Reading Issue #3 of IPSO FACTO, a couple of things caught my attention...

Page 3. The diagram labeled "bus conversion" cannot be expected to work properly. Since a 7404 has totem-pole outputs (i.e. active pullups) it is probably capable of sourcing more current in the high state than the average CMOS output is capable of sinking. Combine this with the (unspecified) pullup resister, and it is unlikely that the CMOS side of the bus will ever get low enough to register a good zero. A 7405 open-collector equivalent (or better the high-voltage 7416) should be used. Furthermore, the logic used to determine bus direction could be either more complex or more simple than that shown, depending on where memory and I/O devices reside. If as implied in the accompanying text, only the CPU is on the CMOS side, then bus direction could be defined adequately by ($\overline{MRD}$+N3), which is high during memory fetch and INP cycles, and low otherwise. If on the other hand you want to allow arbitrary I/O and/or memory on either bus, then the Bus Direction signal must be cognizant of the allocation of memory, and must know whether this memory access refers to CMOS- or TTL-bus memory. In interfacing my 1802 system to an S-100 expansion bus, I chose a middle ground: while I expect both I/O and memory to be on either the TTL or CMOS side, I determined a priori where the CMOS memory is (first 8K) and that I will never access the TTL bus for both I/O and memory (the S-100 spec disallows this anyway), so my Bus Direction is driven by the high three address lines, $\overline{MRD}$, and the N-lines. I will send in a schematic when I am sure it runs. (please do! TC)

With a little more cleverness it is possible to get all 32 (or 64) characters on the TVT-6 line. This is based on the fact that the RTS instruction in the 6502 has a much closer cognate in the 1802 than SEP. The RET instruction is a two-cycle instruction which both does what the SEP does and can be coded to do it with two consecutive memory accesses. Lacking the freedom to specify the second byte (it must be the same as the opcode), this requires that R0, not R14, be used as the main line PC, but this is an inconvenience only to the extent that DMA is disallowed during TVT operation (it would mess up the timing anyway). One further requirement is that X=P during the scan routine, but I do not see that as a major difficulty. It might be simplified if P=7 (since the RET instruction will always set X=7 on exit). If it is desired to preserve D through the scan routine, the XRI instruction (FB) could be used instead of LDI (F8); an even number of these restores the D to its original state (since there are an odd number in the scan routine, one more is required in the main line). Alternately, one OUT instruction could be sacrificed (remember, X=P for the RET), which outputs a constant 6n on the designated port.

For all you guys who want a resident assembler, I have one in the works. It will generate relocatable code, so a relocating loader is included, at what I hope is a reasonable price. It should be ready within a month or so.

Tom Pittman (Itty Bitty Computer Co.), P.O. Box 23189
San Jose, CA, (USA) 95153

Dear Tom:

I am replying to Wayne Bowdish's letter of November 30, 1977, which I received last week (the holiday mail again). Thank you for sending copies of the first three issues of your newsletter: I applaud your enterprising association.

Now of my project. By now you should have read my plea to all CDP 1802 users in the August issue of DDJ. The response has been great but the project is still prenatal. A questionnaire was to have been sent to all respondents and other interested parties this week, but is being delayed pending the outcome of this correspondence.

Two of our biggest problems in forming a "Chicago" club were time and lack of capital to finance a newsletter. If there is anything that we can do from this end, please let me know. Meanwhile please find my application for membership in ACE along with a $5.00 check enclosed.

What will eventually happen is for ACE to become the headquarters for many 1802 Spartan clubs throughout the world. A national meeting is not too far fetched. There are many who are yearning for a user's club. Yet, I can appreciate Milan Skodny's comments in the second issue of IPSO FACTO.

A good point to bring up at your next meeting is how large do you want ACE to become? (Think in particular about finding someone to run this large an enterprise. TC). I think regional clubs who remain autonomous while operating under your banner is an excellent idea.

There are many who, like myself, have taken it upon themselves to generate some interest in the COSMAC system, ie. Ross Wirth, Ed McCormick, H.Shanko, and others. ACE needs representation as well as exposure. Can we consolidate all of these efforts through ACE?

One more item, to Paul Birke of Burlington, Ontario, tell him to contact: Robert Jerald, 103 Coventry Drive, Lakewood, N.J. 08701

He should be finished interfacing the National Semiconductor numbercruncher chip to his 1802 uC.

Yours very truly,
Phil Gennarelli (CDP1802 National Clearinghouse)

Dear Sir:

Thank you for sending information. I would like very much to join your club. Inclosed is my fee.

I have an ELF-II, 1802 based, system which I purchased as a kit from Netronics (advertised in Popular Electronics). I am working on a tape interface to an Alphatype Typesetting machine. It uses reel to reel decks in a saturated digital switching mode and has two channels (one used for a clock track).

Is there anyone out there who knows of this type of recording? (If so, please contact Barney directly. Let me know how you make out, Barney. TC)

I would also like to know where I could buy 86 pin connectors for the ELF-II, also I need a RCA Data Book.

Thank you, Barney Widner, Alpha Service, 24 W Ethel
Lombard, IL 60148

Dear Wayne,

Thanks very much for your letter of Oct. 19th. I was in Canada during Sept. on home leave from a CIDA project that I am working on in Kenya. I am here in a Canadian Govt. aid project and am teaching and writing courses in Microwave technology for the Posts and Telecommunications Dept. of the Kenyan Government. Expect to be here approximately another year and then likely back to Canada where I work for Canadian Pacific Telecommunications.

I attended the IEEE Conference in Toronto during my home leave in Sept. and got "turned on" by the micro-processor explosion that seems to be taking place. I purchased a basic set from Mr. E. Tekatch of Tektron Equip. Which is still not to hand but expected shortly. My background is not entirely lacking in digital techniques but I was most pleased to get your letter as I certainly have lots to learn in the new field of microprocessors. In other words, I am a real beginner and very keen to know more about the construction and uses of this obviously very versatile device. I don't know what I can contribute at this time but would be most happy to have your newsletter and any other information on developments of the RCA 1802. My application and cheque for $5.00 are herewith enclosed.

I am particularly interested in ASCII or BAUDOT interfaces as my connection with CPT could give me access to retired teletypes that could be available at very reasonable cost. Maybe I could help out the club by putting in touch with sources of this sort.
(Definitely! TC)

In the meantime, thanks again and keep the info coming.

Sincerely, Eric Duggan, Box 30305, Nairobi, Kenya, East Africa

Hi,

Just a quick note to let you know I enjoyed the two back issues of the newsletter; they have been very helpful.

I recently built a ELF II 1802 processor, but have not expanded the unit yet. I am an amateur radio operator and build all of my equipment, so I will probably be building some hardware for the ELF.

The editor program and the code program in the newsletter work quite well on the ELF.

Please start my newsletter with issue #3 so I don't miss any articles.

Bob Hillard

Dear Tom,

I have formed a small computer club in the local Junior High School I teach in. We have ordered and are waiting for a 'Super ELF' from Quest Electronics in California - you have probably seen it advertised.At any rate I was most interested to see your notice in the latest BYTE with regard to 1802 users and your plans for memory expansion, I/O, monitor programs and a Basic interpreter. We would like to correspond with you if possible about these developments. Could we be put on your mailing list if one exists? I would be more than willing to defray any costs, etc. Hope to hear from you soon. (You will. Keep in touch. TC)

Yours truly, R. Herbison, RR 2 Horne Lake Road
Port Alberni, B.C., V9Y 7L6

MEMORY-MAPPED I/O for the 1802

Tom Crawford

In the world of micro-processors today, there are 2 main techniques available for I/O to and from peripheral devices. One technique uses special instructions and special hardware signals for input and output. The 8080, Z80 and 1802 micro-processors make this method available, although the implementation is somewhat different in the 1802 as compared to the 8080.

The other technique requires that I/O devices be connected to the CPU as though they were memory elements. The CPU reads from or writes to the device in precisely the same manner as it reads from or writes to memory. The only difference between an I/O device and a memory location is the address. All micro-processors can use this technique; the ones which must use it include the 6800 and 6502.

The 1802 employs 3 I/O lines (N0, N1, N2) and the $\overline{MRD}$ line to implement I/O. The INPx and OUTx instructions (x = 1 to 7) are supplied to control this I/O technique. The hardware connections for up to 3 input ports and 3 output ports are extremely simple: one N-line is assigned to each port, with the state of $\overline{MRD}$ distinguishing between inputs and outputs. Only the following I/O instructions are used: INP1, INP2, INP4, OUT1, OUT2, OUT4. This is the I/O implementation used on the TEK-1802 to handle the on-board keyboard and data LED display under program control.

The addition of a simple 3 line to 8 line decoder to the N-lines allows up to 7 input ports and 7 output ports, and uses the full complement of INP and OUT instructions (note that N0N1N2=000 cannot be used for I/O). This implementation is illustrated in Figure 98 on page 77 of the RCA 1802 USERS MANUAL (MPM-201A).

I am presently planning for some I/O devices I intend to add to my 1802. These include UART's, relay outputs, logic I/O, and analog I/O. I need more than 7 I/O ports. What do I do ?

The 1802 USERS MANUAL shows one possibility, known as 2-level decoding of the N-lines (pages 78 - 79, Figure 99). In my opinion, this method is needlessly complicated, both in the hardware and in the software required to use it.

I intend to implement memory-mapped I/O on my 1802 system. It provides more than enough I/O addressing, and uses the existing memory addressing hardware and software to do this. The only decision required is which memory addresses will be used for memory, and which for I/O. Since the total number of unique memory addresses exceeds 65,000 , there is plenty of space available for I/O devices, without any noticable sacrifice in memory space. Each I/O device simply watches for its unique address on the address lines, then either reads the data lines, or writes to them, depending on the state of $\overline{MRD}$.

Admittedly, watching 16 address lines for an unique combination requires extensive hardware for each I/O device, but this can be simplified considerably if all I/O devices are located in the same region of memory. Then the high order address lines, which are the same for every I/O device, can be monitored by only 1 set of hardware, which produces a single signal which indicates whether any I/O device is being addressed. The same technique can be applied to smaller blocks of devices within the I/O device space, as well.

It is convenient to assign a full memory page for I/O addressing. this allows up to 256 input and 256 output ports, and still leaves 65,279 words of memory. If more I/O devices are required, simply assign 2 pages to I/O devices instead of 1. This allows up to 512 I/O ports.

Figure # 1 shows the hardware required to watch for selection of I/O page #FF by the CPU. The logic produces a signal called IOSEL, which is true if the I/O page is selected. This signal is valid anytime that TPA is false. In particular, it is required to be valid while TPB is true. If you add this signal to your buss structure, then it can be used by any I/O board card you make, but only the first card needs the hardware shown in figure #1. Note that this circuit is guaranteed to work at 5 volts with a CPU clock frequency no greater than 5 MHz (TPA > 200nS).

The circuitry shown in figure #2 is typical of that used to decode individual device selection for several devices on 1 card. It provides 0-true strobe pulses, co-incident with TPB, for up to 4 input and 4 output devices. The 4 device addresses must be adjacent, but may be selected anywhere in the I/O page with 6 jumpers or a DIP-switch connected to the second input of each of the XOR gates.

This selection circuitry is a careful mix of CMOS and TTL logic. The CMOS gates are required to buffer the data from the buss. B-series CMOS is specified, in order to drive 1 TTL input from each CMOS output. The TTL logic is required in order to do the necessary decoding in a reasonable amount of time. The delay time from TPB input to any strobe output is 172 nS max. (90 nS typical). Since data on the bus is valid for 1/2 clock cycle after the end of TPB, then this decoding circuitry is useable with clock frequencies up to 2.9 MHz gauranteed, and up to 5.5 MHz typical. (TTL driven from CMOS B-series is based on experience. For those desiring best noise margins, substitute a 74LS42 for the 7442 IC.)

A similar circuit to provide decoding for 8 inputs and 8 outputs can be made by using a 74154 instead of a 7442. This is shown in Figure #3.

In summary, I have shown a method of mapping I/O ports into addresses, as an alternate to 2-level N-line decoding. I feel that, for more than 7 I/O ports, memory mapping can be implemented with less hardware, and can be used with simpler software, than 2-level N-line decoding.

References

1) USER MANUAL FOR THE CDP1802 COSMAC MICROPROCESSOR, RCA #MPM-201A, pgs 73-79
2) "Ciarcia's Circuit Cellar: Memory Mapped I/O", BYTE magazine, November 1977, pgs 10-16
3) "Build a Universal I/O board", W.T. Walters, KILOBAUD magazine, October 1977, pgs 102-108

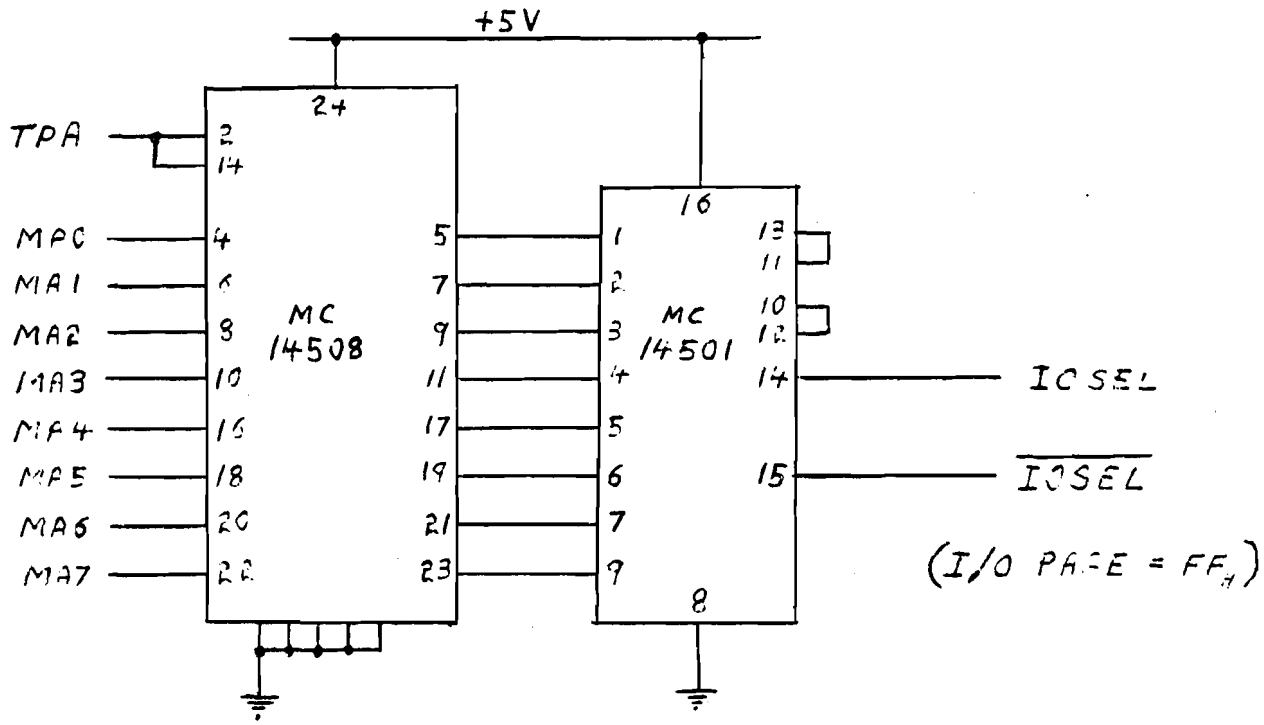


FIGURE #1

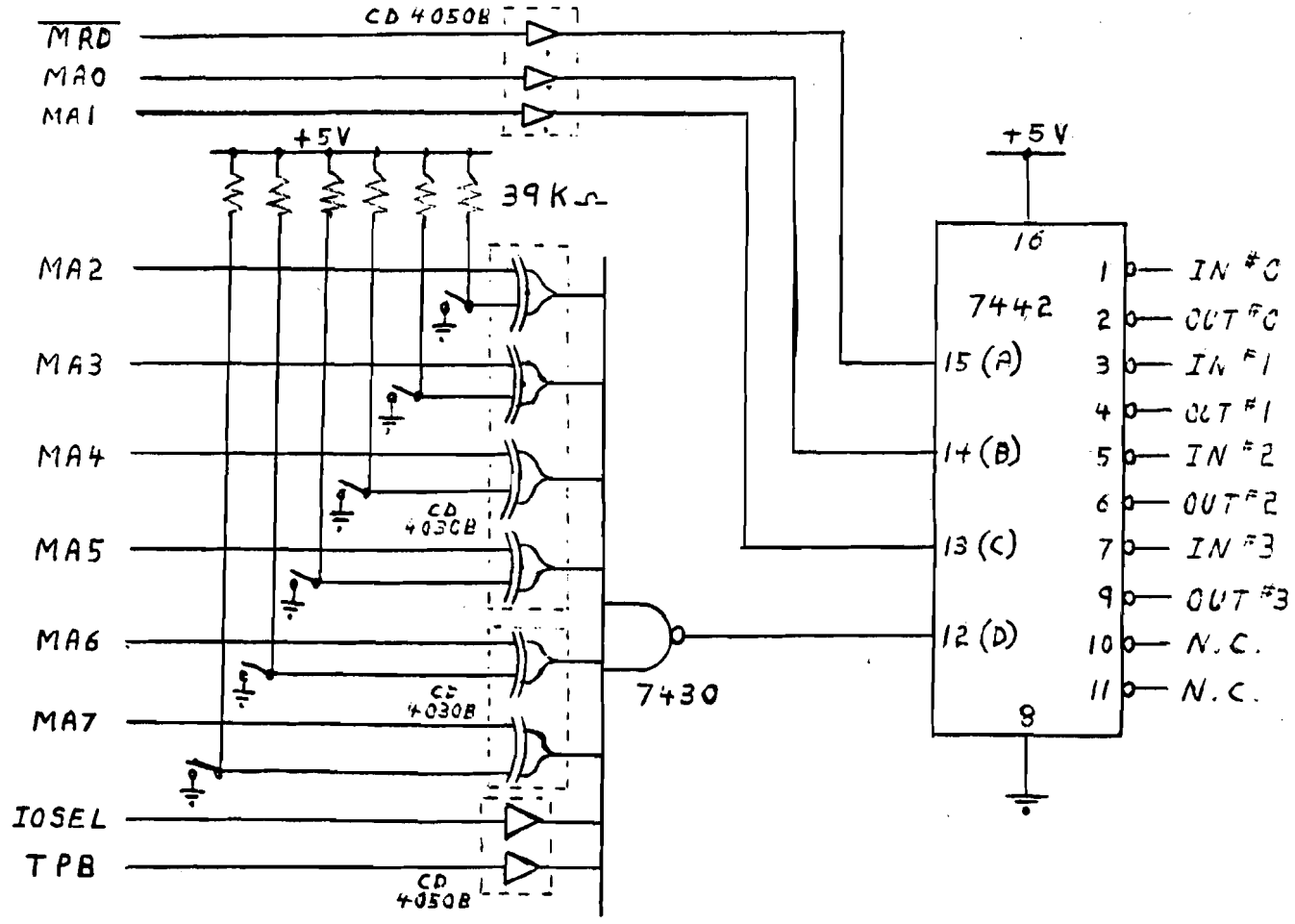


FIGURE #2

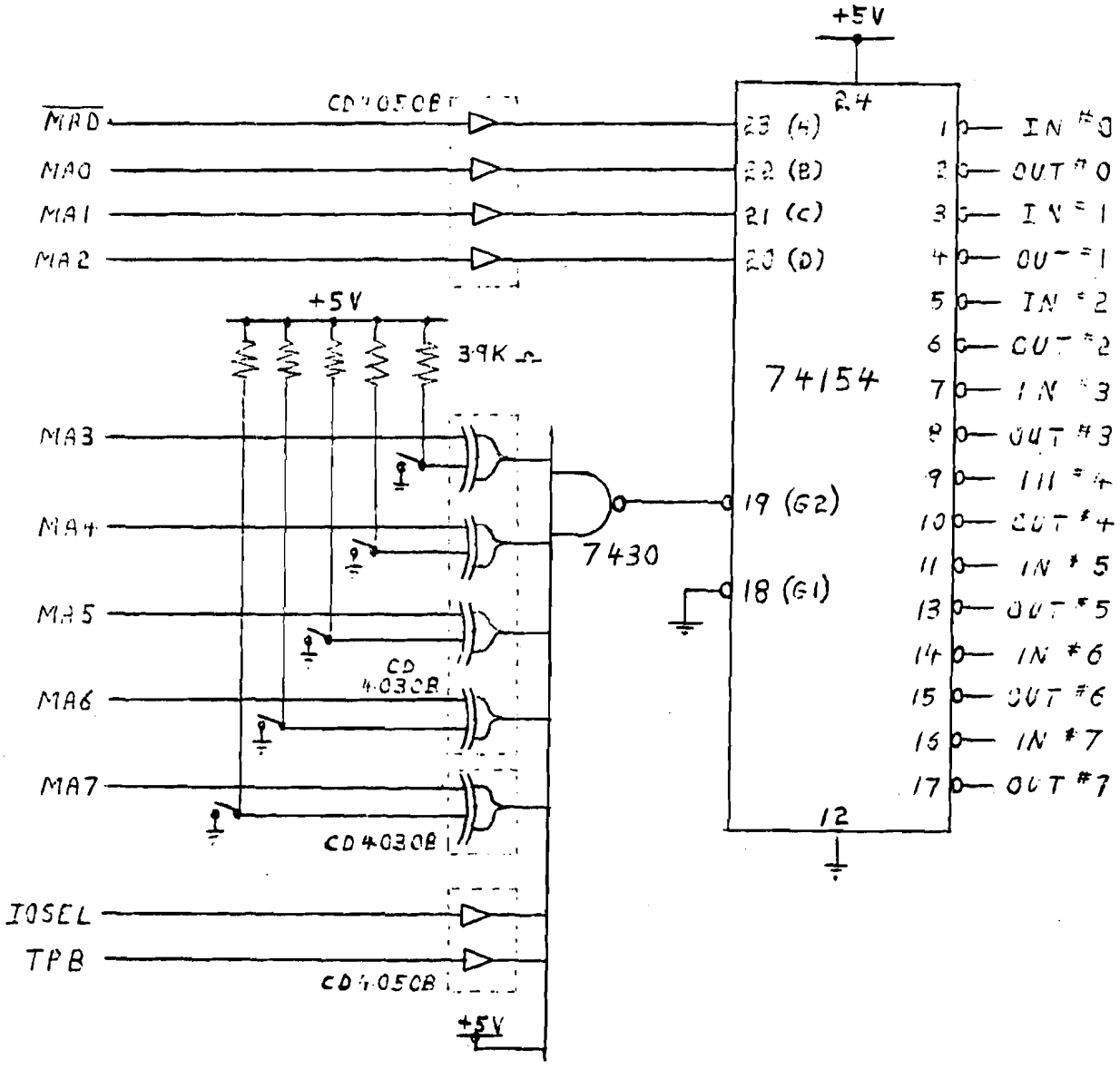

+5V
24
MRD
MA0
MA1
MA2
CD4050B
23 (A)
22 (B)
21 (C)
20 (D)
74154
19 (G2)
18 (G1)
12
1 IN #0
2 OUT #0
3 IN #1
4 OUT #1
5 IN #2
6 OUT #2
7 IN #3
8 OUT #3
9 IN #4
10 OUT #4
11 IN #5
13 OUT #5
14 IN #6
15 OUT #6
16 IN #7
17 OUT #7
+5V
3.9K Ω
MA3
MA4
MA5
MA6
MA7
7430
CD 4030B
IOSEL
TPB
CD4050B
+5V

FIGURE #3

+/- 12 VOLTS ON THE TEKATCH BUS

Tom Crawford

I recently added a Kansas City cassette interface to my micro-processor system. My system is based on the TEC-1802, which employs a 44 pin buss structure. This buss has provisions to supply only +5V power. Also, there are no spare pins on the buss. My KC interface requires +/- 7.5V to supply 2 op amps. In addition, I intend to add a number of other circuits requiring a +ve and -ve supply, such as A-D converters and 3-supply UARTs (cheaper than 5V only UARTs!). I decided that +12V and -12V would serve my purposes and that these should be available on the buss.

Careful examination of the existing pin assignments showed several which are provided for operation of an 1852 I/O Port IC on the CPU board. Since I do not have an 1852 chip, these pins are unused in my system. I therefore made the following changes:

PIN #	OLD USE	NEW USE
2	CS1, $\overline{CS2}$	-12V DC
24	SR	+12V DC
25	CLK(1852)	CLK(1802)

You will note also the change in the assignment of pin 25. This one requires some cutting of traces and a jumper, to connect the 1802 chip to the edge connector on the CPU board. Now you can build an oscillator on another card, and plug it into the buss structure. As a bonus, the CPU clock frequency is available for other purposes, such as Real Time Clock interrupts, a UART time-base, synchronous logic, etc.

I would recommend that you cut the traces on the CPU board which connect edge connector pins 2 and 24 to the I/O Port socket (25 was cut in the previous paragraph), just to avoid potential problems at a later time.

Although I haven't made a pin assignment yet, I have been looking for a pin to hang an IOSEL signal on, for Memory-Mapped I/O. Some thoughts on this subject include: one of $\overline{CE2}$ (pin 43) or CE1(pin 21) can be freed up, leaving only 1 line to select the 256 bytes of memory on the CPU board. A little extra logic will be required on the .75K memory board, to drive the remaining line. Or one could remove the 256 bytes entirely from the CPU board, freeing up 2 lines of the buss. The 2 memory chips can be piggy-backed on 2 chips of the .75K memory board, making it a 1K memory board. It is also possible to free up some unused N-lines or EF lines.

I would be interested in feedback on my +/-12V pin assignments, and on the other comments I have made regarding the Tekatch buss. I emphasize that these are not standard changes, but only modifications to my personal system at this time.

MEMORY TEST ROUTINE FOR THE TEC-1802

A. Dunlop

A recent algorithm by Knaizuk and Hartmann (IEEE Transactions on Computers, April 1977) outlines an ultra-fast RAM test. This algorithm is employed in the test routine shown in Listing 1.

The memory test routine will detect any stuck-at-1 or stuck-at-0 fault in a RAM, including the memory itself, the address and data lines, and the address decoders. It sacrifices testing efficiency to a small degree so as to achieve simplicity. run time is short compared to some other test routines.

Operation of the routine is described in three steps processed for three phases with complemented data used in alternate passes. Each phase is made up of the following three steps:

1. A Data Byte is stored in every location to be tested
2. The complement of the Data Byte is stored in every third location
3. Memory is checked for all locations in the test region

It is important to note that the above three steps must be done as three seperate iterations.

The three steps are performed three times, once for each phase, with the position of the complemented Data Byte changing each time. Then the Data Byte is complemented and the next pass is begun.

The Data Byte is initially #FF giving an FF FF 00 FF FF 00 FF... pattern for the first phase and FF 00 FF FF 00 FF FF... for the second phase and 00 FF FF 00 FF FF 00... for the third phase of the first pass. The second pass first phase generates a 00 00 FF 00 00 FF... pattern.

Memory is tested as a group of bytes between, and including, any two arbitrary addresses allowing small sections of memory to be tested. The low address value of the area to be tested is placed in bytes 2 and 3 of the program at the label 'BEGIN' and the high address limit is placed in bytes 4 and 5 at the label 'END'. Testing progresses from high to low address so as to simplify the limit-checking. The program starts at byte 0 with a branch around the data area to the initialization section.

After 256 complete passes the program will halt unconditionally. The user may wish to NOP the halt instruction to let the test run continuously.

If a fault is detected, the program will halt conditionally with the most significant byte of the fault address displayed in the Data leds, and the Q led indicating what was expected to be in the memory byte at the fault address: Q reset when #00 was expected and Q set when #FF was expected. Pressing and holding the 'I' key will permit the least significant byte of the fault address to be displayed. Upon releasing the 'I' key the program will continue.

The test routine can be executed in any page of memory since it references the absolute page address at initialization time.

This test routine will not solve memory problems or indicate the cause directly. It serves only to direct the user's attention to specific problem areas if they exist.

```
                                .TITLE  --- MEMORY TEST ROUTINE ---
                                        ;MEMORY TEST   RCA 1802   JAN. 1978 AD
                                        ;REGISTER ASSIGNMENT
                                        ;R0     P COUNTER
                                        ;R1     PRESENT BYTE ADDRESS IN MEMORY
                                        ;R2.1   PHASE INDEX (3 PHASES TO TEST)
                                        ;R2.0   INDEX REFLECTING PHASE INDEX
                                        ;R3     ADDRESS OF MEMORY CONTAINING BEGIN
                                        ;R4     ADDRESS OF MEMORY CONTAINING END
                                        ;R5     ADDRESS OF TEMPORARY BYTE
                                        ;R6     NUMBER OF COMPLETE PASSES
                                        ;R7.0   DATA BYTE
                                        ;
                                        ;** CAUTION ** RANGE FOR 'BEGIN'  1 - #FFFF
                                        ;
0000  30 07           START:    BR      INZ     ;
0002  FF FF           BEGIN:    .DBYTE  #FFFF   ;BEGINNING ADDRESS TO BE TESTED
0004  FF FF           END:      .DBYTE  #FFFF   ;ENDING ADDRESS TO BE TESTED
0006  00              TEMP:     .BYTE   0       ;TEMPORARY LOCATION FOR OUTPUT
0007  90              INZ:      GHI     R0      ;INITIALIZE HIGH REG TO
0008  B3                        PHI     R3      ;                       THIS PAGE ADDRESS
0009  B4                        PHI     R4      ;
000A  B5                        PHI     R5      ;
000B  F8 02                     LDI     BEGIN   ;
000D  A3                        PLO     R3      ;R3 SET TO ADDRESS OF BEGIN
000E  F8 04                     LDI     END     ;
0010  A4                        PLO     R4      ;R4 SET TO ADDRESS OF END
0011  F8 06                     LDI     TEMP    ;
0013  A5                        PLO     R5      ;R5 SET TO ADDRESS OF TEMP
0014  F8 00                     LDI     0       ;
0016  B6                        PHI     R6      ;
0017  A6                        PLO     R6      ;R6 SET TO #0000
0018  A7              P000:     PLO     R7      ;DATA BYTE INITIALIZED AS #00
0019  F8 02                     LDI     2       ;SET PHASE INDEX FOR 3 PHASES
001B  B2              P010:     PHI     R2      ;
001C  7A                        REQ             ;RESET Q FOR STEP 1 -  EVERY BYTE
001D  A2              P020:     PLO     R2      ;R2.0 SET FOR "FUNNY SUBTRACT"
001E  44                        LDA     R4      ;
001F  B1                        PHI     R1      ;R1 PRESENT TEST BYTE START AT END
0020  04                        LDN     R4      ;
0021  A1                        PLO     R1      ;
0022  24                        DEC     R4      ;RESET R4 TO POINT TO END AGAIN
0023  39 2C                     BNQ     P040    ;STEP 1 BYPASS
0025  82              P030:     GLO     R2      ;SPECIAL ACTION FOR STEP 2
0026  32 2C                     BZ      P040    ;SUBTRACTION COMPLETE
0028  22                        DEC     R2      ;
0029  21                        DEC     R1      ;"FUNNY SUNTRACT" R1=R1-R2.0
002A  30 25                     BR      P030    ;
002C  13              P040:     INC     R3      ;TO GET LEAST SIGNIFICANT BYTE
002D  81                        GLO     R1      ;                       OF 'BEGIN'
002E  E3                        SEX     R3      ;SET X FOR SUBTRACT 'BEGIN'
002F  F7                        SM              ;
0030  23                        DEC     R3      ;TO GET MOST SIGNIFICANT BYTE
0031  91                        GHI     R1      ;                       OF 'BEGIN'
```

--- MEMORY TEST ROUTINE --- RC1802-V01 MEMTST.RCA 02-FEB-78 17:45:

```
0032  77                        SMB             ;PRESENT ADDRESS  -  BEGIN
0033  3B 43                     BM      P060    ;DONE THIS STEP WHEN DF=0
0035  87                        GLO     R7      ;GET DATA BYTE
0036  FB FF                     XRI     #FF     ;COMPLEMENT IT
0038  39 3F                     BNQ     P050    ;STEP 1 BYPASS
003A  87                        GLO     R7      ;SPECIAL ACTION FOR STEP 2
003B  51                        STR     R1      ;STORE DATA BYTE
003C  21                        DEC     R1      ;DECREMENT PRESENT ADDRESS TWICE TO
003D  21                        DEC     R1      ;BYPASS NEXT TWO BYTE ADDRESSES
003E  38                        SKP             ;
003F  51              P050:     STR     R1      ;STORE  COMPLEMENTED DATA BYTE
0040  21                        DEC     R1      ;DECREMENT PRESENT ADDRESS
0041  30 2C                     BR      P040    ;
0043  31 49           P060:     BQ      P100    ;FINISHED STEP 2
0045  7B                        SEQ             ;SET Q FOR STEP 2 - EVERY THIRD BYTE
0046  92                        GHI     R2      ;GET PHASE INDEX
0047  30 1D                     BR      P020    ;GO TO STEP 2 ENTRY
0049  44              P100:     LDA     R4      ;STEP 3 TEST FOR CORRECT DATA
004A  B1                        PHI     R1      ;
004B  04                        LDN     R4      ;
004C  A1                        PLO     R1      ;R1 SET TO END ADDRESS
004D  24                        DEC     R4      ;RESET R4 TO POINT TO END AGAIN
004E  92                        GHI     R2      ;GET PHASE INDEX
004F  A2                        PLO     R2      ;SET FOR INDEX TO EVERY THIRD BYTE
0050  13              P110:     INC     R3      ;                       FOR THIS PHASE
0051  81                        GLO     R1      ;
0052  E3                        SEX     R3      ;SET X FOR SUBTRACT 'BEGIN'
0053  F7                        SM              ;
0054  23                        DEC     R3      ;
0055  91                        GHI     R1      ;
0056  77                        SMB             ;PRESENT ADDRESS  -  BEGIN
0057  3B 7E                     BM      P200    ;DONE STEP 3 WHEN DF=0
0059  82                        GLO     R2      ;
005A  32 61                     BZ      P120    ;BRANCH TO TEST THIRD BYTE
005C  87                        GLO     R7      ;
005D  FB FF                     XRI     #FF     ;COMPLEMENT DATA BYTE
005F  30 65                     BR      P130    ;
0061  F8 03           P120:     LDI     3       ;
0063  A2                        PLO     R2      ;RESET R2.0 TO TEST THIRD BYTE
0064  87                        GLO     R7      ;                       FROM HERE
0065  CE              P130:     LSZ             ;
0066  7B                        SEQ             ;SET  Q  IF D=#FF
0067  38                        SKP             ;
0068  7A                        REQ             ;RESET Q  IF D=#00
0069  E1                        SEX     R1      ;SET X FOR EXCLUSIVE OR
006A  F3                        XOR             ;COMPARE WITH MEMORY BYTE
006B  32 7A                     BZ      P140    ;OK
006D  E5                        SEX     R5      ;SET X FOR OUTPUT FAULT ADDRESS
006E  91                        GHI     R1      ;
006F  55                        STR     R5      ;
0070  64                        OUT     4       ;OUTPUT MOST SIGNIFICANT BYTE OF
0071  25                        DEC     R5      ;                       FAULT ADDRESS
0072  3F 72                     BN4     .       ;WAIT FOR '1' PUSHED
0074  81                        GLO     R1      ;
```

```
--- MEMORY TEST ROUTINE ---      RC1802-V01      MEMTST.RCA    02-FEB-78  17:45:53      PAGE   3

 1  0075  55                     STR     R5      ;
 2  0076  64                     OUT     4       ;OUTPUT LEAST SIGNIFICANT BYTE OF
 3  0077  25                     DEC     R5      ;                          FAULT ADDRESS
 4  0078  37 78                  B4      @       ;WAIT FOR 'I' RELEASED
 5  007A  22             P140:   DEC     R2      ;R2.0=R2.0-1
 6  007B  21                     DEC     R1      ;R1=R1-1
 7  007C  30 50                  BR      P110    ;
 8  007E  92             P200:   GHI     R2      ;
 9  007F  A2                     PLO     R2      ;
10  0080  22                     DEC     R2      ;DECREMENT PHASE INDEX
11  0081  3A 8C                  BNZ     P210    ;PHASE INDEX NOT -1, DO NEXT PHASE
12  0083  16                     INC     R6      ;INCREMENT COUNT OF PASSES
13  0084  96                     GHI     R6      ;
14  0085  3A 85                  BNZ     @       ;STOP AFTER COMPLETING 256 PASSES
15  0087  87                     GLO     R7      ;
16  0088  FB FF                  XRI     #FF     ;COMPLEMENT DATA BYTE FOR NEXT PASS
17  008A  30 18                  BR      P000    ;GO DO ANOTHER PASS
18  008C  82             P210:   GLO     R2      ;GET DECREMENTED PHASE INDEX
19  008D  30 1B                  BR      P010    ;GO DO NEXT PHASE
20                               .END            ;
```

```
 --- MEMORY TEST ROUTINE ---      RC1802-V01      MEMTST.RCA    02-FEB-78  17:45:53      PAGE   4
USER SYMBOL TABLE

BEGIN  0002   END    0004   INZ    0007   P000   0018   P010   001B   P020   001D   P030   0025   P040   002C
P050   003F   P060   0043   P100   0049   P110   0050   P120   0061   P130   0065   P140   007A   P200   007E
P210   008C   START  0000   TEMP   0006

          PROGRAM SIZE = 008F
            0 ERRORS DETECTED

*
```

OTHER CLUB NEWSLETTERS

The following list shows the publications produced by other clubs, which we have obtained, often in exchange for IPSO FACTO. To see any of these, simply contact any member of the ACE Executive at the next ACE meeting.

COSMAC ELF NEWSLETTER (Charles Manry, 2012 Williamsburg Ct. S., Leaque City, Texas 77573)
- November 77 (3 pages)
- December 77 (3 pages) - includes a reprint from IPSO FACTO
- January 78 (3 pages)

MCMASTER UNIVERSITY COMPUTER CLUB (Hamilton, Ontario)
- Introductory Information. September 11 (1 pages)

TRIANGLE AMATEUR COMPUTER CLUB (PO BOX 17523, Raleigh, N.C. 27609)
- August 77 (2 pages)
- Sept. 77 (2 Pages)

(This club seems to have disappeared. TC)

TULSA COMPUTER SOCIETY (PO BOX 1133, Tulsa, Oklahoma 74101)
- September 77 (4 pages)
- October 77 (4 Pages)

RANDOM NUMBER GENERATION FOR THE 1802

B.J. Murphy

Many computer applications require the use of random numbers. Computer games such as STAR-TREK, TIC-TAC-TOE use random number routines to set up initial conditions.

You may be asking yourself "How can a computer generate a truly random number?" The fact of the matter is that computers cannot generate random numbers - the computer can be programmed however, to generate a "pseudo-random" number.

Before you turn the page to read the next article, have a look at listing 1. This 1802 routine will generate random numbers from 0 to 255. The program however, always generates the same sequence of random numbers; thus we call this sequence "pseudo-random".

By changing the initial value of the variable "RANDOM", (statement 40) the program generates different sequences of random numbers. This initial value is formally known as the random number seed.

An interesting technique is the use of the summation of powers of two to achieve multiplication. The equation in statement 9 of the program can be changed to allow the multiplication of any number.

If you wish to learn more about random number generation, consult the reference or any good Computer Science textbook.

REFERENCE: Grieser "Pseudorandom Number Generator", Byte Magazine, November 1977, page 218.

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.2

0000              1 ************************************************************
0000              2 ***                                                        ***
0000              3 ***    PSEUDO-RANDOM NUMBER GENERATOR FOR RCA1802         ***
0000              4 ***                                                        ***
0000              5 ***                                                        ***
0000              6 *** METHOD:  TAKE PREVIOUS RANDOM NUMBER,MULTIPLY BY 13   ***
0000              7 ***          AND ADD 1                                     ***
0000              8 ***                                                        ***
0000              9 *** MULTIPLY BY 13 BY USING: 13*N = N*2**3 + N*2**2 + N   ***
0000             10 ***                                                        ***
0000             11 *** REGISTER USAGE: R0=P.C.; R2=X REG ; R3,R4=WORK         ***
0000             12 ***                                                        ***
0000             13 ************************************************************
0000             14 *
0000 F8 18       15 MAINLOOP LDI   RANDOM        GET ADDR OF PREVIOUS RANDOM #
0002 A2          16          PLO   R2            SAVE INTO R2
0003             17 *        CLEAR R2 HIGH FOR SYSTEMS OVER 256 BYTES
0003 E2          18          SEX   R2            MAKE R2 X REG
0004 F0          19          LDX   R2            GET PREVIOUS RANDOM #
0005 FE          20          SHL                 X2
0006 FE          21          SHL                 X2 AGAIN
0007 A3          22          PLO   R3            SAVE N*2**2
0008 F4          23          ADD                 D= N + N*2**2
0009 52          24          STR   R2            STORE THE RESULT
000A 83          25          GLO   R3            GET N*2**2
000B FE          26          SHL                 X2 AGAIN GIVES N*2**3
000C F4          27          ADD                 D= N + N*2**2 + N*2**3
000D FC 01       28          ADI   1             FINISH BY ADDING 1
000F 52          29          STR   R2            STORE RESULT
0010             30 *
0010             31 *       END OF RANDOM NUMBER CODE GENERATION
0010             32 *
0010 64          33           OUT4               DISPLAY RANDOM #
0011 B4          34           PHI   R4           USE AS INTERVAL
0012 24          35 BUZZ      DEC   R4           DECREMENT
0013 94          36           GHI   R4           GET THE VALUE
0014 3A 12       37           BNZ   BUZZ         DONE ?
0016 30 00       38           BR    MAINLOOP     YES..ANOTHER RANDOM #
0018             39 *
0018 39          40 RANDOM    DC    57           RANDOM NUMBER SEED
0019             41 *
0019             42           END
   0 DIAGNOSTICS GENERATED
   3 SYMBOLS
SYMBOL TABLE:
MAINLOOP     0000
BUZZ         0012
RANDOM       0018
READY
```

ROM OUT OF RAM

M. Pupeza

For those with the TEC 1802 3/4K Memory Expansion board, there is an easy way to write-protect the CMOS memory page, making it look for all intents and purposes like ROM. All that is required is a switch, 2 resistors, an optional LED indicator, and some wiring to existing spare gates in IC3 and IC4. The wiring is shown in Figure 1 below. Be sure to cut the existing traces which connect pin 20 of IC7 and IC8 to the $\overline{MWR}$ line, when you connect up the memory protect circuit.

With the switch in the "Normal" position, the CMOS memory operates as RAM. In the "Protect" position, the memory cannot be written into at all, hence looks like ROM. As described in the Expansion board notes, 2 "AA"-size batteries will maintain the contents of your "ROM" when you turn off the power to the rest of the system.

(Ed. Note: If you do write to this block of memory when it is Write-Protected, both the CPU and the memory drive the Data buss. This could cause trouble. TC)

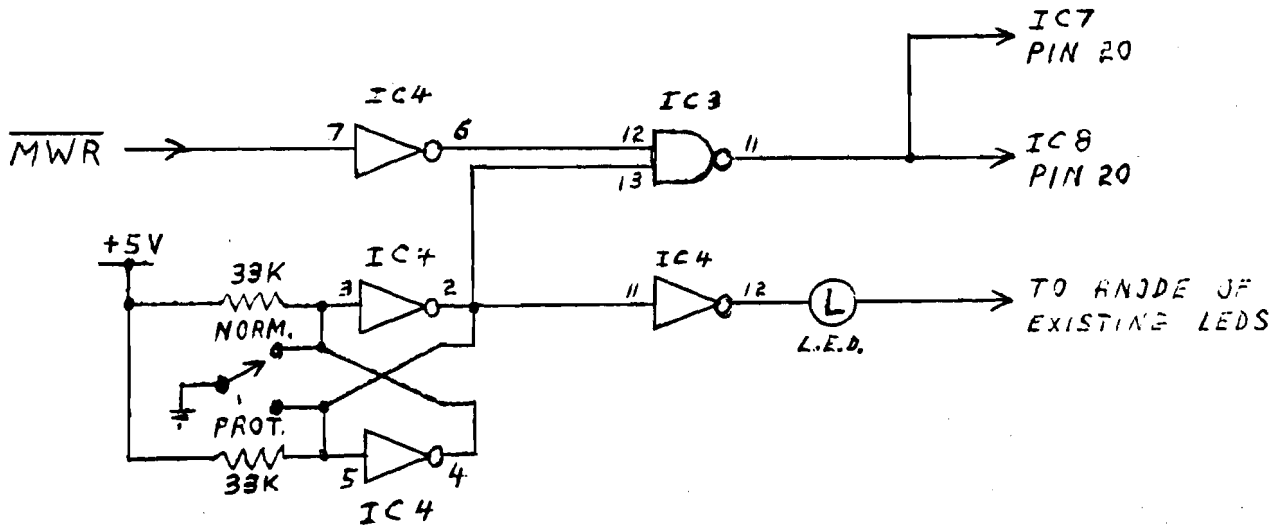


FIGURE 1

ITEMS FOR SALE

COSMAC ELF (as seen in P.E. August 76) with
- built in 5V supply, 31 pin dual I/O connector.
- enclosed in box (RS #270-232)
- Hex keyboard controller for hex input and remote control functions (no software needed)
- speaker with transistor amp.
- asking $75. Contact Richard Kindig,
712 Mitchell Rd., Clearfield, PA, 16830
Phone: (814) 765-4873

CAPACITORS, silver mica, for sale. All values (1-3000pf), $.20 each. Call Gerr Waite, (416) 385-5491, or write 150 Mohawk Rd. E. Apt #617, Hamilton, Ont., L9A 2H1.

ANOTHER KEYBOARD APPROACH

Richard Kindig

Here's some information on my HEX keyboard; the unit plugs directly into my ELF with no modifications except the addition of a switch to disconnect the original input switch. My ELF has 3 position input switches, having a center (none) position. Without this, the switches would either have to be ORed with the keyboard (all switches off) or removed from service with some sort of a bus seperator.

Encoding 16 keys to hexadecimal can be done several ways. The ways I've used include the use of a HD0-165 encoder chip, and a circuit discussed in a Popular Electronics article (How to Fully Debounce Low Cost Keyboards, January 1977, page 51). Using the

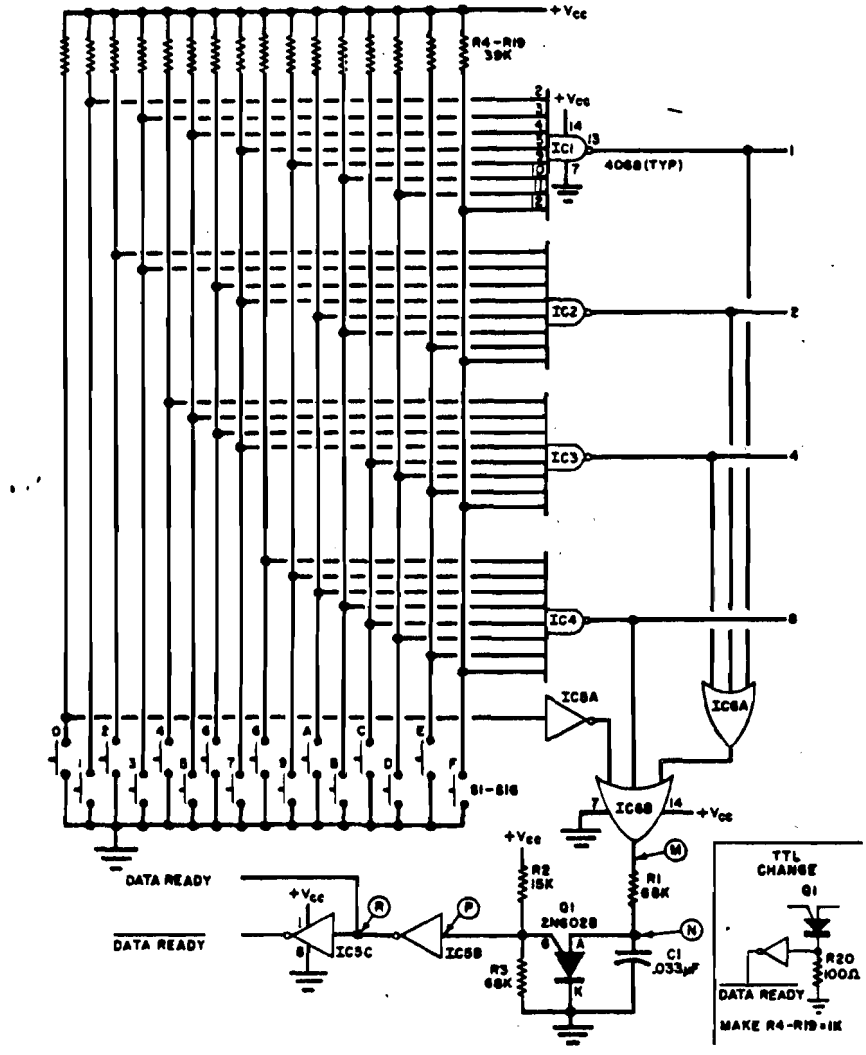


Fig. 1. The circuit produces a 1-2-4-8 binary code and provides a debounced data-ready strobe.

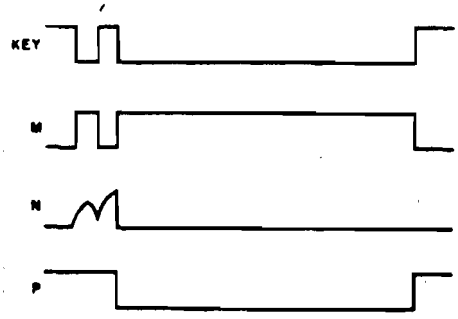


Fig. 2. Waveforms show how bouncing key can produce a clean single output.

PARTS LIST

C1—0.033-μF, 15-V capac tor
IC1-IC4—CD4068 8-input NAND gate (for TTL, use 7430)
IC5—CD4049 inverting hex buffer (for TTL, use 7404)
IC6—CD4075 triple 3-input OR gate (for TTL, use 7432)
Q1—2N6028 programmable unijunction transistor
R1,R3—68,000-ohm, ¼-watt resistor
R2—15,000-ohm, ¼-watt resistor
R4-R19—39,000-ohm, ¼-watt resistor
R20—100-ohm, ¼-watt resistor (TTL only)
S1-S16—Normally open switch/keyboard

encoder chip is very easy since it provides internal pulldown resistors, debouncing and a key-pressed signal. The device however is somewhat expensive, and has a dissipation of 88 millamperes. (presently my ELF with hex displays and keyboard uses slightly more than 200 ma.). The P.E. circuit which I'm now using, has C-MOS chips and a unijunction transistor (for debouncing). (see figures 1 and 2).

With the encoded output, it is then necessary to latch a high and low order digit sucessivily for each key pressed. To do this, I routed the Key-pressed signal to a flip-flop for binary division. (see figure 3 below) The Q and Q outputs are then connected to half-monostable circuits, to produce a pulse that would latch an input before the key is released. These pulses are strobed to the store pins of 4042 latches, and the output is then held on the Q output lines (which are connected to the center position of the input switches). The Q output of the flip-flop is also buffered and connected to an LED, which indicates whether the high or low digit will be entered next.

An input switch and LOAD, MEMORY-PROTECT, and RUN switches were mounted on the same board, resulting in a remote, hand-held controller, connected to the ELF by a 16 conductor ribbon cable. When using the keyboard, care must be taken that none of the ELF's switches are out of their neutral position.

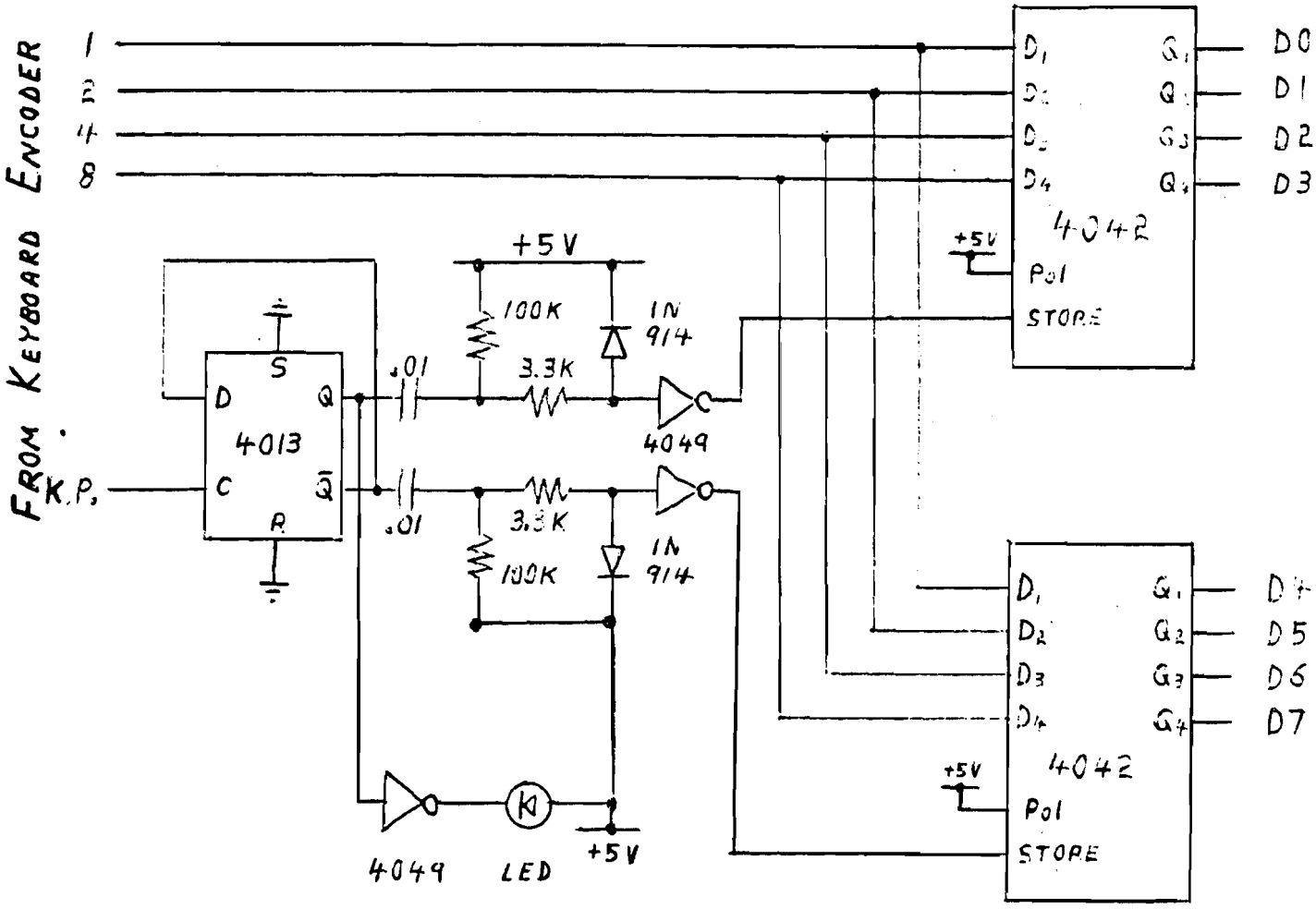


FIGURE 3

INTERUPT PROCESSING ON THE 1802

Bernie Murphy

This article deals with a flaw in the interupt architecture of the RCA 1802. If the reader does not understand the concept of interupts, he/she should consult the references. The BYTE article is highly recommended.

Having mastered all the basic 1802 instructions, I thought that the interupt oriented instructions should be tested. I was shocked when I discovered that the 1802 does not have an interupt disable instruction! Page 65 of the RCA1802 USER MANUAL (MPM-201m) states " The RETURN and DISABLE instructions can be used to set or reset IE ... A convenient method is to set X equal to the current P value and then perform the RETURN (70) or DISABLE (71) instruction using the desired X,P for the immediate byte."

For example: assume X=2, P=3

```
                .
                .
                .
E3      SEX     R3      set X=3
71      DIS             return to X=2, P=3, IE=0
23      #23             immediate byte
                .
                .
                .
```

The RCA manual forgets to mention that if an interupt occurs just after the SEX R3 instruction, the machine code preceeding the DIS instruction becomes the data stack for the interupt routine, not a very desirable situation.

A possible solution to this problem is to add some extra hardware to the interupt system and use a latched output such as the Q line to control interupts. (see figure 1)

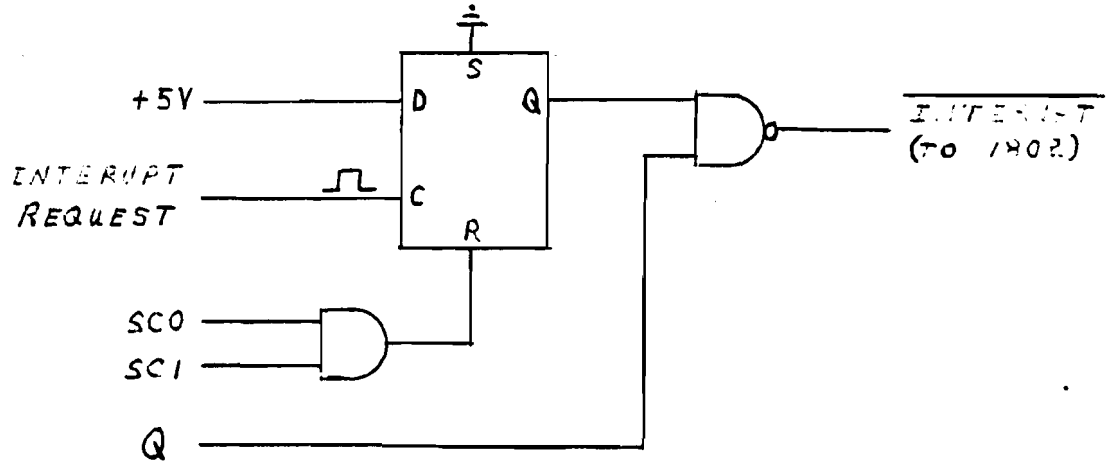


FIGURE 1

The sequence to provide "critical processing" would be:

```
REQ             disable interupt gate
 .
 .              do critical instruction sequence
 .
SEQ
```

If anyone has a software solution for the above mentioned problem please contact the editor (or write an article !).

A simple program that demonstrates interupt processing is found in listing 1. Try it - short the interupt line to ground and see what happens...

If one desires to "fake an interupt" via software, the following code will do the trick:

```
DEC     R2              decrement stack
MARK                get X,P into T register
SEX     R2          reset X register again
INC     R2          increment stack ( because of MARK )
INC     R2          increment stack
SEP     R1          fake interupt now...
```

The above method is very useful when debugging interupt routines. Just think, the Motorola 6800 does all of the above with one instruction, SWI (Software Interupt) !

REFERENCES

1. "A LITTLE BIT ON INTERUPTS", Byte, December 1977, p.188
2. USER MANUAL FOR THE CDP1802 COSMAC MICROPROCESSOR, MPM201A, RCA CORPORATION
3. COSMAC Microprocessor Product Guide, MPG-180, RCA Corporation, p.6
4. IBM 360 Programming and Computing by Golden and Leichus, Prentice-Hall Inc.

ERRATTA

There are 3 errors in the Tape Test Routine found on page 30 of Issue #3 of IPSO FACTO:

PC	OLD INSTR.		NEW INSTR.	
1A	7A	REQ	7B	SEQ
28	7B	SEQ	7A	REQ
46	3D	BN2	35	B2

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.

0000             1 ************************************************
0000             2 *** 1802 DEMO PGM SHOWING INTERUPT PROCESSING  ***
0000             3 ***                                            ***
0000             4 ***       R1    INTERUPT P.C.                  ***
0000             5 ***       R2    STACK (X REG)                  ***
0000             6 ***       R3    MAIN TASK P.C.                 ***
0000             7 ***                                            ***
0000             8 ************************************************
0000 F8 00       9           LDI   0              ZERO D REG
0002 B1         10           PHI   R1             CLEAR HIGH BYTE
0003 B2         11           PHI   R2             OF ALL REGS USED
0004 B3         12           PHI   R3
0005 E2         13           SEX   R2             R2 IS X REG
0006 F8 24      14           LDI   INTERUPT       ADDR OF INTERUPT ROUT
0008 A1         15           PLO   R1             SET UP IN R1
0009 F8 FF      16           LDI   STACK          ADDR OF STACK
000B A2         17           PLO   R2             SET UP IN R2
000C F8 10      18           LDI   MAINTASK       ADDR OF MAIN TASK
000E A3         19           PLO   R3             FOR P.C.
000F D3         20           SEP   R3             CHANGE P.C.
0010 F8 07      21 MAINTASK  LDI   7              PATTERN 3 LIGHTS
0012 A7         22           PLO   R7             SAVE INTO R7
0013 64         23 MAINLOOP  OUT4                 OUTPUT PATTERN
0014 22         24           DEC   R2             FIX UP STACK
0015            25 *
0015            26 *
0015 F8 04      27           LDI   4              MAIN TASK DELAY
0017 B8         28           PHI   R8             SAVE IT
0018 28         29 MAINDLY   DEC   R8             COUNT DOWN
0019 98         30           GHI   R8             LOAD INTO D
001A 3A 18      31           BNZ   MAINDLY        DONE ?
001C 87         32           GLO   R7             GET PATTERN
001D 7E         33           SHLC                 SHIFT LEFT
001E A7         34           PLO   R7             SAVE AWAY
001F 52         35           STR   R2             NEW PATTERN
0020 30 13      36           BR    MAINLOOP       DO IT AGAIN
0022            37 *
0022            38 ************************************************
0022            39 ***       INTERUPT ROUTINE (RUNS DISABLED)     ***
0022            40 ************************************************
0022 42         41 INTERRTN  LDA   R2             RESTORE D
0023 70         42           RET                  RETURN TO CALLER
0024 22         43 INTERUPT  DEC   R2             DEC STACK
0025 78         44           SAV                  SAVE X,P ONTO STACK
0026 22         45           DEC   R2             DEC STACK
0027 52         46           STR   R2             SAVE D REG
0028 7B         47           SEQ                  TURN ON Q
0029 F8 10      48           LDI   $10            DELAY
002B BF         49           PHI   R15            SAVE IT
002C 2F         50 INTERDLY  DEC   R15            COUNT DOWN
002D 9F         51           GHI   R15
002E 3A 2C      52           BNZ   INTERDLY       DONE ?
0030 7A         53           REQ                  TURN OFF Q
0031 30 22      54           BR    INTERRTN       DONE WITH INTERUPT
0033            55 *
00FF            56 STACK     ORG   $FF            STACK AREA
00FF            57 *
00FF            58           END
   0 DIAGNOSTICS GENERATED
   7 SYMBOLS
SYMBOL TABLE:
MAINTASK    0010
MAINLOOP    0013
MAIN!
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                          1802 VER 1.3

                  1 ************************************************
                  2 ***                                          ***
                  3 ***        RCA 1802 MINI EDITOR              ***
                  4 ***     (THIS VERSION FOR FULL SYSTEM)       ***
                  5 ***                                          ***
                  6 ***        WRITTEN BY   E. TEKATCH           ***
                  7 ***       DIS-ASSEMBLED BY B. MURPHY         ***
                  8 ***                                          ***
                  9 ***  REGISTER USAGE:  R15=PC                 ***
                 10 ***                   R14=WORK & PC IN RUN MODE***
                 11 ***                   R13=SUBROUTINE CALL ADDR ***
                 12 ***                   R12=X REG               ***
                 13 ***                   R11-R2=FREE TO USE      ***
                 14 ***                   R1=WORK REG             ***
                 15 ***                   R0=INITIAL PC           ***
                 16 ***                                          ***
                 17 ***  MAINTENANCE RECORD:                     ***
                 18 ***  11/12/77    PHI R14 ADDED               ***
                 19 ***  12/03/77    OFF Q FOR RUNMODE ADDR      ***
                 20 ***                                          ***
                 21 ************************************************
0000 F8 68       22 RESTART  LDI   BUFFER1       ADDR OF KEYBOARD DATA
0002 AC          23          PLO   R12           PLACE INTO R12
0003 F8 54       24          LDI   KEYIN         KEYBOARD IN ROUTINE
0005 AD          25          PLO   R13           SET UP ADDR IN R13
0006 F8 69       26          LDI   BUFFER2       ADDR OF KEYBOARD DATA
0008 AE          27          PLO   R14           SET INTO R14 LOW
0009 F8 1E       28          LDI   MAINLINE      ADDR OF MAIN CODE
000B AF          29          PLO   R15           PUT IN R15 LOW
000C F8 00       30          LDI   00            INIT D REG
000E B1          31          PHI   R1            ZERO
000F B2          32          PHI   R2              ALL
0010 B3          33          PHI   R3                HIGH
0011 B4          34          PHI   R4                 ORDER
0012 B5          35          PHI   R5                   OF
0013 B6          36          PHI   R6                     REGS...
0014 B7          37          PHI   R7
0015 B8          38          PHI   R8
0016 B9          39          PHI   R9
0017 BA          40          PHI   R10
0018 BB          41          PHI   R11
0019 BC          42          PHI   R12
001A BD          43          PHI   R13
001B BE          44          PHI   R14
001C BF          45          PHI   R15
001D DF          46          SEP   R15           CALL MAINLINE
001E 7B          47 MAINLINE SEQ               ON Q-INDICATE INSTRUC
001F FC          48          SEX   R12           X=ADDR(BUFFER1)
0020 DD          49          SEP   R13           CALL KEYBOARD IN
0021 6C          50          INP4                GET INSTRUC REQUEST
0022 3A 48       51          BNZ   RUNMODE       BRANCH IF RUN MODE
0024 7A          52          REQ                 OFF Q-ADDR REQUEST
0025 EE          53          SEX   R14           X=ADDR(BUFFER2)
0026 DD          54          SEP   R13           CALL KEYBOARD IN
0027 6C          55          INP4                GET HIGH BYTE OF ADDR
0028 BE          56          PHI   R14           PUT IN R14 HIGH
0029 DD          57          SEP   R13           CALL KEYBOARD IN
002A 6C          58          INP4                GET LOW BYTE OF ADDR
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                          1802 VER 1.3

002B AE          59            PLO   R14          PUT IN R14 LOW
002C 64          60            OUT4               OUTPUT DATA AND INC X
002D C4          61            NOP                DON'T KNOW WHY ????
002E 7B          62 INSTRUCT   SEQ                ON Q-INDICATE INSTRUCTION
002F EC          63            SEX   R12          X=ADDR(BUFFER1)
0030 DD          64            SEP   R13          CALL KEYBOARD IN
0031 6C          65            INP4               GET REQUEST
0032 EE          66            SEX   R14          X=ADDR OF DATA TO MODIFY
0033 3A 3B       67            BNZ   NOTZERO      BRANCH IF NOT DATA MOD
0035 7A          68            REQ                OFF Q-DATA REQUIRED
0036 DD          69            SEP   R13          CALL KEYBOARD IN
0037 6C          70            INP4               GET DATA AND ZAP STORAGE
0038 64          71            OUT4               DISPLAY DATA AND INC X
0039 30 2E       72            BR    INSTRUCT     NEXT INSTURCTION
003B FD 01       73 NOTZERO    SDI   1            SUBTRACT 1
003D 3A 42       74            BNZ   NOTONE       BRANCH IF NOT 1
003F 64          75            OUT4               INC X AND DISPLAY DATA
0040 30 2E       76            BR    INSTRUCT
0042 FC 01       77 NOTONE     ADI   1            ADD 1 TO INSTRUC TYPE
0044 3A 6A       78            BNZ   NOTTWO       BRANCH IF NOT TYPE 2
0046 2E          79            DEC   R14          BACK UP ONE M.A.
0047 2E          80            DEC   R14          1 MORE BECAUSE OF OUT4
0048 64          81            OUT4               OUTPUT DATA AND INC X REG
0049 30 2E       82            BR    INSTRUCT
                 83 *
004B 7A          84 RUNMODE    REQ                OFF Q -INDICATE ADDR
004C DD          85            SEP   R13          CALL KEYBOARD IN
004D 6C          86            INP4               GET DATA ADDR HIGH
004E BE          87            PHI   R14          PLACE IN R14 HIGH
004F DD          88            SEP   R13          CALL KEYBOARD IN
0050 6C          89            INP4               GET DATA ADDR LOW
0051 AE          90            PLO   R14          PLACE IN R14 LOW
0052 DE          91            SEP   R14          CALL PGM AND HOPE ...
                 92 *
0053 DF          93 KEYRTN     SEP   R15          RETURN TO MAINLINE
0054 3F 54       94 KEYIN      BN4   *            WAIT TILL I PRESSED
0056 F8 05       95            LDI   5            DELAY VALUE
0058 B1          96            PHI   R1           SET UP DELAY
0059 21          97 KEYDLY1    DEC   R1           COUNT DOWN
005A 91          98            GHI   R1           GET RESULT INTO D
005B 3A 59       99            BNZ   KEYDLY1      BRANCH IF NOT DONE
005D 37 5D      100            B4    *            DEBOUNCE I BY
005F F8 05      101            LDI   5            ANOTHER DELAY
0061 B1         102            PHI   R1
0062 21         103 KEYDLY2    DEC   R1
0063 91         104            GHI   R1
0064 3A 62      105            BNZ   KEYDLY2      BUZZ TILL DONE
0066 30 53      106            BR    KEYRTN       SET UP RETURN
                107 *
0068 00         108 BUFFER1    DC    0            KEYBOARD DATA BUFFER
0069 00         109 BUFFER2    DC    0            KEYBOARD DATA BUFFER
                110 *
006A FC 01      111 NOTTWO     ADI   1            ADD 1 TO INSTRUCTION
006C 3A 00      112            BNZ   RESTART      NOT 0,1,2...SO RESTART
006E 7A         113            REQ                OFF Q-DATA REQUIRED...
006F DD         114 MODIFY     SEP   R13          CALL KEYBOARD IN
0070 6C         115            INP4               GET DATA AND ZAP STORAGE
0071 64         116            OUT4               OUT DATA AND INC X REG
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT                    1802 VER 1.3

0072 30 6F      117           BR     MODIFY        DO IT FOREVER...
                118 *
0074            119           END

   0 DIAGNOSTICS GENERATED
  14 SYMBOLS

                      BUFFER1   0068  KEYRTN    0053  INSTRUCT   002E
SYMBOL TABLE:         BUFFER2   0069  KEYIN     0054  NOTZERO    003B
RESTART      0000     NOTTWO    006A  KEYDLY1   0059  NOTONE     0042
MAINLINE     001E     MODIFY    006F  KEYDLY2   0062  RUNMODE    004B
```

NOTES ON CONNECTING AN 1861 TO A TV

Barney Widner

I hooked my CDP1861 - GRAPHICS CHIP up to a Panasonic color portable set. I wired a jack as shown in figure 1 to the 1st video amp but my video was reversed. I had to build the kluge shown in figure 2.

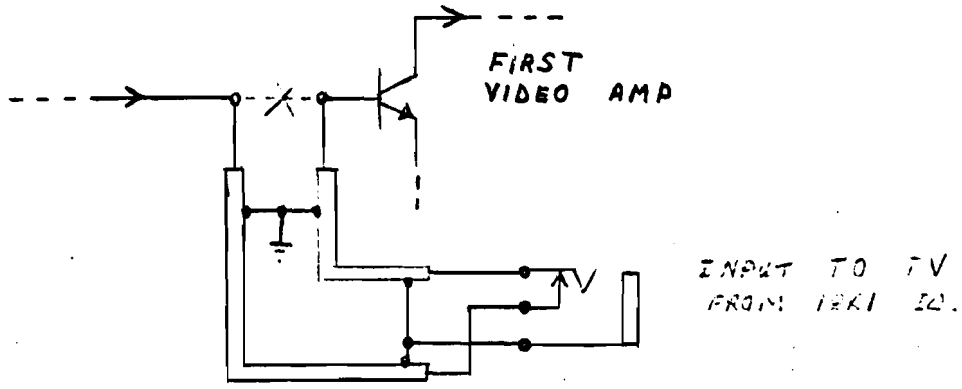


FIGURE 1

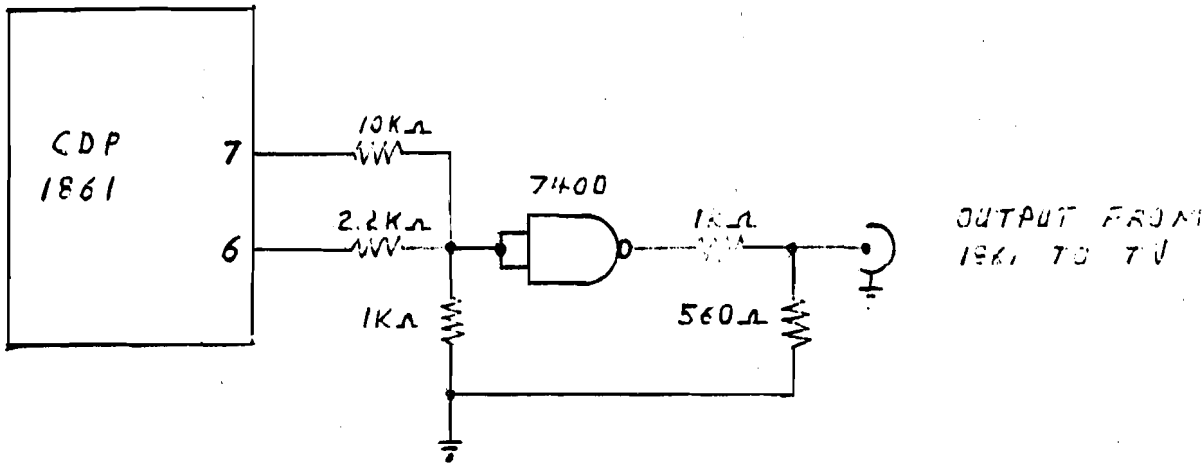


FIGURE 2

THE ASSOCIATION OF COMPUTER EXPERIMENTERS
MINUTES OF CLUB MEETING NO. 78-1

Held at the Spectator Auditorium 10-JAN-78 8:00 pm

78-1-1 The regular meeting was preceded by a 1 hour tutorial conducted by Ken Smith. The tutorial concept was well received and will continue on a twice-a-month basis.

78-1-2 The President, Tom Crawford, announced that the dues collected should last until the end of the fiscal year (May 31). The annual dues will be reassessed at that time.

78-1-3 Motion to adopt Minutes 77-3:
Proposed - George York
Seconded - John Norris
Carried unanimously.

78-1-4 George York, Secretary-Treasurer reported a current bank balance of $812.58. There had been an expenditure of $100. advanced to Tom Crawford to cover Newsletter and president's expenses.

After discussion it was decided to include the minutes of the meetings as part of the Newsletter.

78-1-5 Membership Co-ordinator Claus Doerwald was absent due to illness. Wayne Bowdish reported there are about 190 paid members to date and 75 information packages have been sent out to interested parties.

78-1-6 Software Co-ordinator, Wayne Bowdish, made a request for software articles for the Newsletter. These would be broken into 2 areas of interest: general interest e.g. games, light flashers, random number generators; and system software e.g. operating systems for 1802, monitor programs, etc. Also there is still interest in cassette interfaces, video and keyboard interface systems and software, memory test routines, etc.

78-1-7 Hardware. Tom Crawford reported the Tektron memory system is nearly ready for production.

78-1-8 Education Co-ordinator, Ken Smith, reported the tutorials will continue on a twice-a-month basis. The format will be kept informal.

78-1-9 Newsletter Editor, Tom Crawford, reported the next Newsletter should be out near the end of January. Thanks to the generous support of Dofasco and IBM the cost of the last Newsletter was about $.40/issue, mainly for postage and envelopes.

78-1-10 New Business - Lyle Sandy has volunteered to develop and handle an information exchange system for the club. Samples of a typical data sheet were distributed for comments.

The possibility of group purchases arranged by the executive was discussed. The executive will investigate the situation.

The sale of kits (e.g. 1861 display of memory map, Kansas City interface board, terminal board) by the club was discussed. Brian Fox will investigate the possibility of obtaining a terminal board kit for about $75.00 if the volume is adequate.

Accepting paid advertising to cover the cost of the Newsletter was discussed. The concensus was agreeable but our limited circulation probably makes it uneconomic for a single advertiser.

Announcements of bad weather meeting cancellations will be made on CHML about 6:00 PM.

78-1-11 The next meeting will be Thursday, February 9, 1978 in the Stelco Wilcox St. auditorium. Tutorial at 7:00 PM, regular meeting and flea market at 8:00 PM. There will also be a tutorial evening in the Stelco auditorium February 23 at 8:00 PM.

78-1-12 Motion to adjourn meeting:
Proposed - Tom Crawford
Seconded - John Norris
Passed
The meeting adjourned about 10:00 PM. About 50 people attended the meeting.

Unless notice to the contrary, the following meeting/tutorial schedule is in effect. All meetings and tutorials at the Stelco Wilcox St. auditorium.

Tutorial/Meeting (7 PM/8 PM)	Tutorial Only (8 PM)
Thursday, Feb. 9	Thursday, Feb. 23
Tuesday , Mar. 7	Tuesday , Mar. 21
Thursday, Apr. 6	Thursday, Apr. 20
Tuesday , May 9	Tuesday , May 23

1802 INFORMATION SHEET

At the December meeting it was suggested that we should collect data on members' systems. As a result, we have included on page 36 an Information Sheet for each member to fill out and return to Lyle Sandy, who will collect the results and make them available at club meetings. This will allow members to determine who else is working ona particular project, so as to minimize effort required on each project. In addition, Lyle will attempt to summarize the data, for presentation as an article in the next issue of IPSO FACTO.

*

Information Sheet for 1802 System Belonging To: _____.

I wish to exchange information:
- at club meetings (Y-N) ____
- also at home. Phone ___-____
- also at work. Phone ___-____

	OPERATING	UNDER CONSTR.	NEAR FUTURE	EVENTUALLY
MEMORY-				
Rom (specify amt and chip type)				
Ram " " " "				
Prom " " " "				
C-MOS " " " "				
Audio cassette (specify recording standard)				
Digital cassette (specify make)				
Floppy disc (specify make)				
Other (specify)				
SOFTWARE-				
Monitor				
Editor (Tek. Version 1.3)				
Editor (other)				
Assembler				
Cross Assembler (which computer?)				
Basic Interpreter				
RCA VIP routines				
Other (specify)				
HARDWARE & INTERFACING-				
Faster clock for 1802 (specify frequency)				
Higher voltage on 1802 (specify voltage)				
Real-time clock (specify frequency)				
T.V. terminal				
Ascii terminal				
Baudot terminal (specify make-model)				
Hex display				
Motherboard (Tek)				
Motherboard (other)				
S-100 bus adapter				
Relays				
A/D, D/A conversion				
Stepping motors				
Opto isolators				
Power failure protection				
Programmable counter/timer				
Uart/Modem				
1852 - 8 bit I/O Port				
1853 - N bit decoder				
1854 - Uart				
1856/7 - 4 bit buffer				
1858/9 - latch				
1861 Video/graphics				
4049/4050 TTL buffers				
Interface to other computers (specify which)				
Other interfaces or hardware (specify)				
APPLICATIONS-				
Ham radio				
Games				
Industrial				
Business				
Alarm systems				
Music generator				
Medical				
Other (specify)				

HELP PROMOTE INFORMATION EXCHANGE. FILL OUT THIS SHEET AND MAIL TO— LYLE SANDY, 637 GLEN MOOR CR. BURLINGTON ONT. L7N2Z8

This page contains a change of address form and an application for membership form. If your address is incorrect or if your address has changed, please mail in the change of address form. This will ensure that you get your next copy of the IPSO FACTO Newsletter. Don't forget to return your old mailing lable. This will allow us to find the incorrect address and correct it.

If you know of anyone who would be interested in joining our club, why not give him/her the membership application. We are constantly looking for new members with new and interesting ideas. Make cheques payable to the Association of Computer Experimenters.

Association of Computer Experimenters

CHANGE OF ADDRESS FORM

ATTACH MAILING LABLE HERE

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE

MEMBERSHIP APPLICATION FORM FOR THE Association of Computer Experimenters

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE