IPSO FACTO

Issue *2
Sept 1977

Table of Contents

Editor, ipso facto : Tom Crawford
Invaluable Assistants: Jo-Ann Van Bergen, Wayne Bowdish, Eugene Tekatch, and all contributors to this Issue.

GREAT EXPECTATIONS

There are 2 items of importance to discuss in this issue:

1) The formation of a Micro-processor Club.
2) The future existence of this Newsletter.

The first item evolves from discussion with several people concerning the advantages of a club. Its principle value is in the exchange of ideas and information required in the development of a working micro-computer system. The task of single-handedly building a useful computer from scratch is monumental. If the task can be split amongst a number of like-minded people, however, the situation becomes more practical.

Another advantage concerns help. If you have a tricky circuit problem, or a software bug you can't find, bring it (them?) to a club meeting. Chances are, you will find someone there who can help.

Now, on to the future existence of this newsletter. To date, funding this newsletter has been a joint effort of Tektron Equipment Corp. and various scroungers. A firm financial footing is required at this point. In addition, the several authors involved in the first 2 issues would like to see alot more group participation (i.e. write some articles!). You don't have to be an expert in your field to write for this newsletter. All you need is an idea, a pencil, and a piece of paper! You don't even have to have a working application; if you've been mulling over some idea, put it down on paper and send it in; perhaps someone can help you out with some practical suggestions; or perhaps they have already done it, and can tell you how to get yours working. If you have a working idea, then for goodness' sake tell the rest of us how you did it!

At this point, I would like to announce a meeting to discuss both of the above points, among others. All interested people are invited; you don't need a micro-processor to come to the meeting. A proposed agenda is shown on the following page, along with the where and when. There will also be a number of working micro-computers and peripheral devices on display. These will all be home-built pieces of equipment, and will demonstrate what can be done with time and ingenuity, and only a little money.

Micro-processor Club Meeting

Place: Dofasco Sherman Centre, North Room (basement)
1047 Barton Street East, Hamilton

Time : Wednesday, Sept. 21, 1977, 8 P.M.

Proposed Agenda:

1) Introduction
2) Presentation of Results of Questionaire
3) Discussion of Results
4) Show of interest in Club and/or Newsletter
5) Nomination and Election of Officers of Club
6) Suggestions for future meetings
7) Introduction of Demonstrations
8) Demonstrations & Discussions
(Coffee will be served after the formalities)

You will have noticed that Item #2 on the Agenda is the presentation of the results of a Questionaire. In order to have some results to present, I would like every interested reader to fill out the Questionaire on page 4 of this Newsletter, and send it to me, at the address shown. The results of the Questionaire will be used as guidelines for operation of a Micro-processor Club, should we form one. Please try to mail the form before Sept. 10, 1977; otherwise, there won't be time to prepare the results for the meeting.

I hope to see most of you at the meeting!

Niagara - St. Catherines Computer Club?

I would also like to note that there seems to be some interest in forming a Computer Club in the Niagara - St. Catherines area. Interested people should contact Mr. John Clark, Dept. of Electronics, Niagara College, Welland.

1802 Course offered again ...

The 1802 Micro-Processor course is being offered again, starting October 12, 1977. The course is sponsored by the Hamilton Section of the IEEE. For more information, contact Mr. Tony Wallace, 196 Homewood Ave., Hamilton, Ontario. Phone residence 526-6154 or business 388-0240 Ext. 511.

IEEE 1977 Conference and Exposition

... is being held at the Automotive Bldg, CNE Grounds, Toronto, September 26-28, 1977. You can get in free if you fill out and send the enclosed advance registration card. Further details can be obtained from Mr. Eugene Tekatch, 957-7556, or from any IEEE member. (P.S. You don't have to be an IEEE member to go).

"Letters" and "Items for Sale"

There are 2 new Sections in this issue. I hope they will continue to appear, and to grow, in future issues. When writing for these Sections, please be brief, and type your letters, if at all possible. Thanks.

In Closing

I would like to say Thanks to all the people who have supplied articles for this Newsletter. I think the response has been quite good, and I hope it continues for future issues.

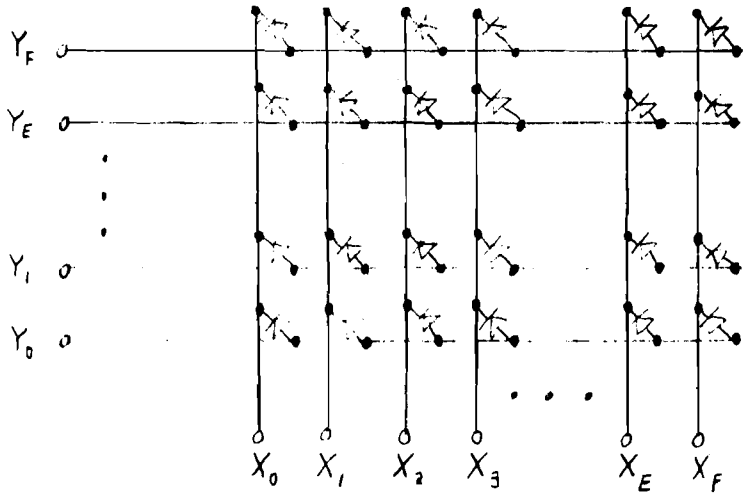
---

16 X 16 L.E.D. Matrix

Buster Waldock

The following information shows how to build a general purpose LED display matrix for the 1802 Micro-processor. This device connects to the Data Bus and the N2 Output line, and is addressed with an OUT4 (64) instruction.

The matrix of light-emitting diodes (LEDs) is arranged as shown:



"GREAT EXPECTATIONS" QUESTIONAIRE

Send to: TOM CRAWFORD
S.M. & G. ELECTRICAL DEPT.
DOFASCO, P.O. BOX 460
HAMILTON, ONTARIO
L8N 3J5

1) I am interested in a micro-processor oriented:
   a) Club only b) Newsletter only c) Both d) Neither

2) I would pay annual dues for the above of:
   a) None b) $2 c) $5 d) $10

3) I would attend club meetings:
   a) Never b) Bi-weekly c) Monthly d) Bi-monthly

4) I would like to see a newsletter published:
   a) Never b) Bi-weekly c) Monthly d) Bi-monthly

5) I would contribute articles to a newsletter:
   a) Never b) Occasionally c) Frequently d) Every issue

6) I consider my electronic hardware ability:
   a) Non-existent b) Beginner c) Fair d) Good

7) I consider my programming ability:
   a) Non-existent b) Beginner c) Fair d) Good

8) If asked, I would assist others with their problems:
   a) Never b) Occasionally c) Frequently d) Anytime

9) I presently own or have access to a micro-processor:
   a) No b) 1802 c) 8080 d) Other (Please specify) _____

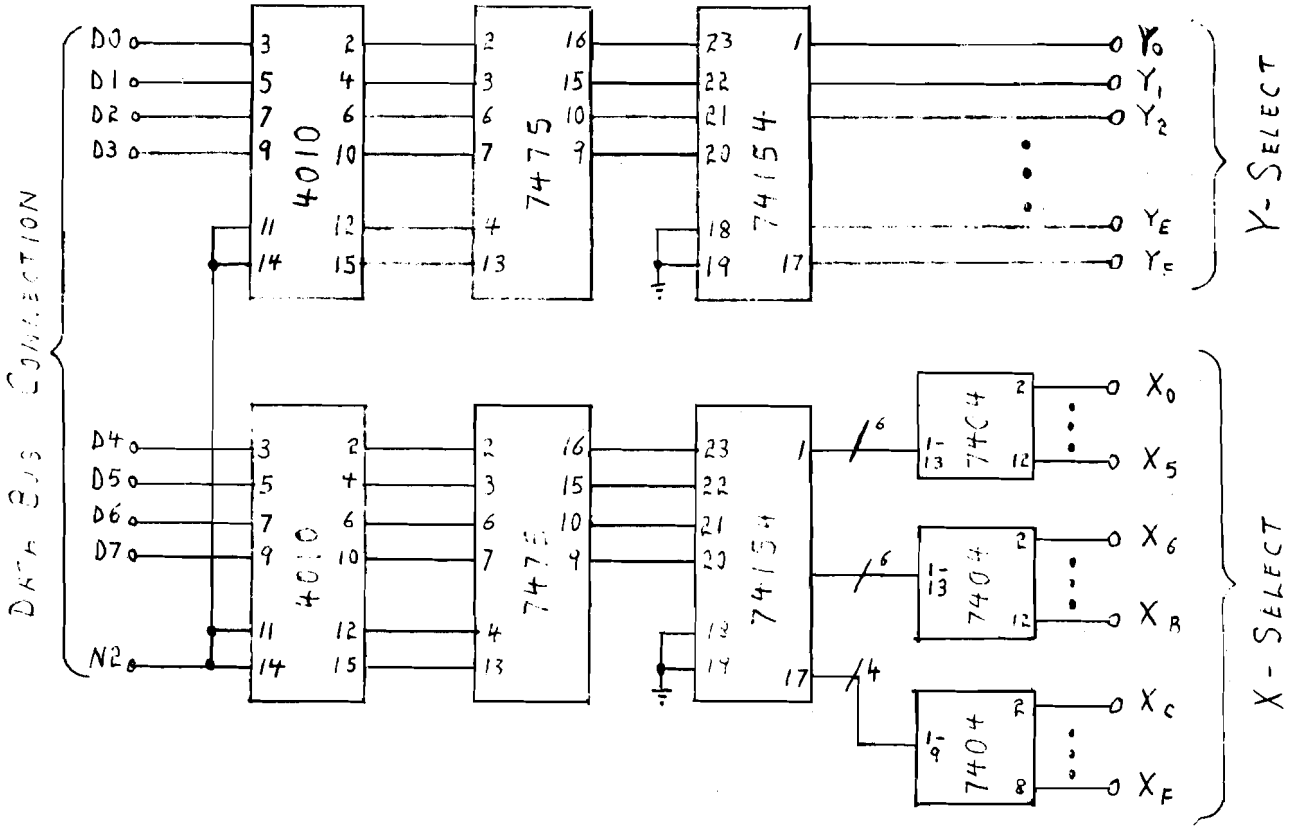
10) I am willing to actively assist with a club/newsletter:
   a) Yes b) No

Comments:

(This page is left blank intentionally, to allow you to remove and send in the Questionaire on the other side of this sheet. TC)

By connecting 1 of the Y-axis lines to ⏚ , and simultaneously connecting 1 of the X-axis lines to +5V (via a current limit), a particular LED can be turned on.

The particular arrangement chosen to do this, consists of 2 4-bit latches, and 2 4 line to 16 line decoders. In this way, 1 8 bit byte can be used to select a particular LED, by storing the Y address in bits 0-3, and the X address in bits 4-7. To protect the Data Bus from being loaded down, two 4010 chips are used to buffer the signals from the Data Bus, and 3 7404s (hex inverters) are used to invert the 16 lines from 1 of the decoders, in order to provide a +5V TRUE selection along the X axis. (The decoders normally provide 0 TRUE decoding). The arrangement of hardware looks as follows:



The software required to drive the display can be coded in a number of ways. The simplest approach is to store a pre-determined set of X - Y addresses in a table, then output the data from the table in a sequential manner. When the end of the table is reached, go back to the beginning and start through the table again. (If you don't repeat the outputs, your picture will appear momentarily, then disappear).

A sample program, which displays a square in the lower left corner of the display, is given below:

```
Memory
Address        Instruction          Comment

  00             F8 15          Set up start address of table
  02             A4                 in R4
  03             F8 0D          Set up # of LED's +1
  05             A3                 in R3
  06             23
  07             83
  08             32 00
  0A             F8 XX          Delay time (any value from 00 to
                                                        FF)
  0C             A1
  0D             21
  0E             81
  0F             3A 0D
  11             E4
  12             64
  13             30 06
  15             22             Table starts here
  16             32
  17             42                          7 00000000
  18             52               Display:   6 00000000
  19             23                          5 00XXXX00
  1A             53                          4 00X00X00
  1B             24                          3 00X00X00
  1C             54                          2 00XXXX00
  1D             25                          1 00000000
  1E             35                          0 00000000
  1F             45
  20             55             end of table    01234567
```

For further information and more test routines, contact the author. Budget limitations make it necessary to omit some excellent routines using this display.

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)

To the Editor:

Here are some comments, questions and suggestions:
The Mother Board has -5V and -12V markings. Does this mean that we will have to buy more power supplies?
I quote: "The existence of this newsletter will depend upon your interest and participation." Do you keep in mind that most of the nearly 300 kit owners have only a 6 X 9 PC Board loaded with components, and a battery from Canadian Tire as an option? And might be working hard to get a subtraction routine going? Get down to ground. Sophisticated software and fancy hardware are still far ahead for most of us. How about small hardware improvements for starters? ...
Most kit owners have the slow CD chip. Any advice for them for building "Music and Micros"?
Have you considered a "Dear Abby" column for the kit owners having software or hardware problems?
... Isn't it worth considering adding hex readouts as an option? Its faster and less prone to errors. After all, we don't write programs in binary ... One of the kit owners has already built the hex readout.
Have you considered adding a Proto-board compatible with the 1802 kit hardware/mother board?
I have enclosed an ad for a reasonably priced Paper Tape reader (74.50) which might be of use to some readers (Oliver Audio Eng. Model OP-80A. TC) Maybe someone can design something similar just for us.

Sincerely, Milan Skodny
Port Dover, Ontario

(You have brought up some excellent points. Some of your questions are answered in this issue; perhaps you can supply answers yourself to some of the others. I hope to see you at the meeting. You show an active interest in micro-processors.TC)

Dear Eugene:

Thank you for sending me the 1st edition of the "Tec News Letter". It was very interesting and I hope that this edition will only be the 1st of many. Perhaps sometime in the future, when I become more familiar with the 1802, I will be able to submit something for publication ....

Regards, Geff Waite
Hamilton, Ontario

(Thank you for the compliment, Geff. Please don't wait too long to submit that article, though. We don't need experts, just interest. TC)

Dear Eugene:

I must thank you and Dr. G. Carter for an excellent course. I enjoyed it very much. My TEC-1802 works well and must be considered the bargain of the year at $85.

... I would like to make a suggestion concerning ... a BASIC interpreter written ... around an on-board number-cruncher calculator chip. In this way little software effort and memory space would be expended in order to implement the ordinary (most used) algebraic and trig functions.

Enclosed is some info on the National Semi-conductor "Number Cruncher" calculator chip (see ELECTRONICS Magazine, Feb. 17, 1977 page 103 to 106. TC) ... The marriage of this chip (and its 2 FIFO's) to the TEC-1802 is in my opinion, a top priority project ... I know that Mr. George Mychailenko of Brantford is trying to make it work with the TEC-1802.

Another project which should be added to the Hardware list is an ASCII Keyboard/TVT (including PIX) using full 80 characters X 16 lines ... .

Thanks again; Paul V.Birke
Burlington, Ontario

(Excellent ideas! Perhaps I can convince you and George to write up your hybrid Interpreter for the Newsletter? Also, I refer you to BYTE Magazine, September and October, 1976, for details on a similar Mathematical Function Unit, and to BYTE, August and November 1976, for several excellent construction articles on Video displays, from the simplest to the most complex, for both alphanumerics and graphics. TC)

A CROSS ASSEMBLER - WHAT'S IT MAD AT??

In our first newsletter, it was stated that a cross assembler was being written for the 1802. Well, it's true. In fact, it's almost ready. To anyone familiar with software, the advantages of an assembler are obvious. Those of you who are not software oriented (yet!) may ask, What is a cross assembler and why have one? Well here it is -

WHAT?

A cross assembler is a program. It runs on a computer other than your 1802. (Many of you will have access to computers through either your company or a friend). Its purpose is to generate the machine code which you will later enter into your 1802's memory.

The cross assembler accepts your 1802 program (called the source program) in a symbolic form (via punched cards, a terminal, etc.). By symbolic form, I mean that you express instructions and addresses using symbolic labels or codes. For example FB,17 might become XRI BCDMSK and 33,5F might become BGE TO.BIG .

The machine code which the assembler produces may be on several different mediums. For instance, the assembler might print a list of the HEX codes which you then enter through your HEX keyboard. If you have a card reader hooked up to your 1802 (a future issue - we hope - will contain an article on how to build a cheap card reader), the assembler could punch the binary data directly onto cards. If you decide to add RØM to your 1802, why not let the assembler drive your PRØM programmer?

WHY?

But if I don't have a card reader or PRØM programmer, why waste time entering my program into the assembler? I will have to enter the hex codes through the keyboard anyway. Well, that's true, but there are still some advantages to using the assembler. Here is a partial list:

- symbolic codes (opcodes) and labels are easier for humans to use and comprehend. For instance, suppose you have the following sequence of bytes in a program: 04, B5, 25, 95, 3A, 25. I've seen programs published in magazines like this. What does it do? Is it a data table or executable code (instructions)?

The equivalent assembly language program would look like this:

```
        LDN R4
        PHI R5
DELAY:  DEC R5
        GHI R5
        BNZ DELAY
```

After studying the code for a few seconds, it becomes apparent that it is a delay loop. There was never any question about it being a data table. With the addition of comments, the intent of the code would be even clearer.

- programs are more easily modified. For instance, you have a 200 byte program and wish to insert 5 instructions in the middle. If all you have is a list of the hex codes, you must go through each instruction looking for branch instructions. If you find one, the address byte(s) may have to be changed. While you're at it, you had better check for any addresses which you are loading into registers since they probably changed too. Using the assembler, all memory locations which you reference are given symbolic labels (DELAY in the above example). The assembler worries about putting the correct address byte(s) into the instructions.

- an assembler can simplify the programmers' job. You can probably remember, or at least guess, the action performed by the instruction ADD but what's the hex code for it? The assembler has a good memory for this type of thing. Here is another example, you have a routine for outputting messages (maybe you've built a TV Typewriter, or found an old teletype). Now let's code a message. Get out your ASCII to hex conversion card and enter -

  4D,59,20,44,4F,47,20,48,41,53,20,46,4C,45,41,53

  The message is obvious, isn't it? Here is how to do it with the assembler.

  .ASCII /MY DOG HAS FLEAS/

  The .ASCII is called a pseudo-op. It simply instructs the assembler to generate a series of bytes containing the codes for the characters between the delimiters (the "/" in this case).

This briefly describes the what and some whys for a cross-assembler. If you would like to see some articles on assemblers, why not write me a note, or better yet, come to the meeting on September 21.

The real intent of this article was to inform you of the fact that a cross assembler is being written for the 1802. It should be ready in the fall. It has been written in FORTRAN so that it should run on most machines with a minimal number of changes. Any BASIC type out there could probably convert it. I will make it available to anyone who is interested for the cost of copying the listing (probably $3 or $4).

Wayne Bowdish
149 East 33rd St.
Hamilton, Ont.
L8V 3T5

P.S.

This newsletter would provide an excellent format for trading enhancements and additions to the assembler (and informing others of bugs). This is true for any software or hardware projects. Tutorial type articles (like how do assemblers work? or how do A/D and D/A converters work?) are another possibility. Why not lend it some support. At least fill out the questionaire in this issue. Also, come to the meeting on September 21. Let's hear from you!

Items for Sale

1) Computer System for Sale: IMSAI 8080 $3300.00
   - Front panel, 22 slot mother board, 28 amp power supply, 24K lo-power static RAM, 2K EPROM board, hi-speed video interface, 1200 baud cassette interface, parallel I/O card, serial EIA interface, 9" video monitor, paper tape reader, 5K and 8K BASIC interpreters, Assembler, Text Editor, etc.

2) Texas Instruments Silent 700 KSR-33 Terminal $1200.00

3) Teletype ASR-33 with paper tape reader and punch $ 750.00

   - For more information on the 3 above items, contact

   Mr. Arun Rele
   P.O. Box 4364
   Hamilton, Ontario L8V 4L8
   Phone 388-6863 or 385-2762

4) Light Dependent Resistors (LDR's) $ .10 each
   - Dark resistance is 3MΩ min., light resistance is 7KΩ max. at 13 fc. No response times are given. Peak spectral response is at 600nm.

5) Walnut vinyl covered open wooden wraps $ .75 each (plus 4 feet and metric mounting screws). Outside dimensions are 8.125" W * 3.5" H * 9.0" D. Wood thickness is 0.25" approx.

6) Master Standard Oscillator for organ $ 75.00 (tuning fork standards). Two of the 12 tone standards are missing, but the notes can be obtained by dividing down a higher standard. Also included is a 7 octave divider board.

   Items #4, 5, and 6 are offered by Mr. Patrick Anthony. He can be reached at 239 Ottawa St. N., Kitchener, Ont. N2H 3K9
   Residence phone (519) 743-3366 'til 8 P.M.

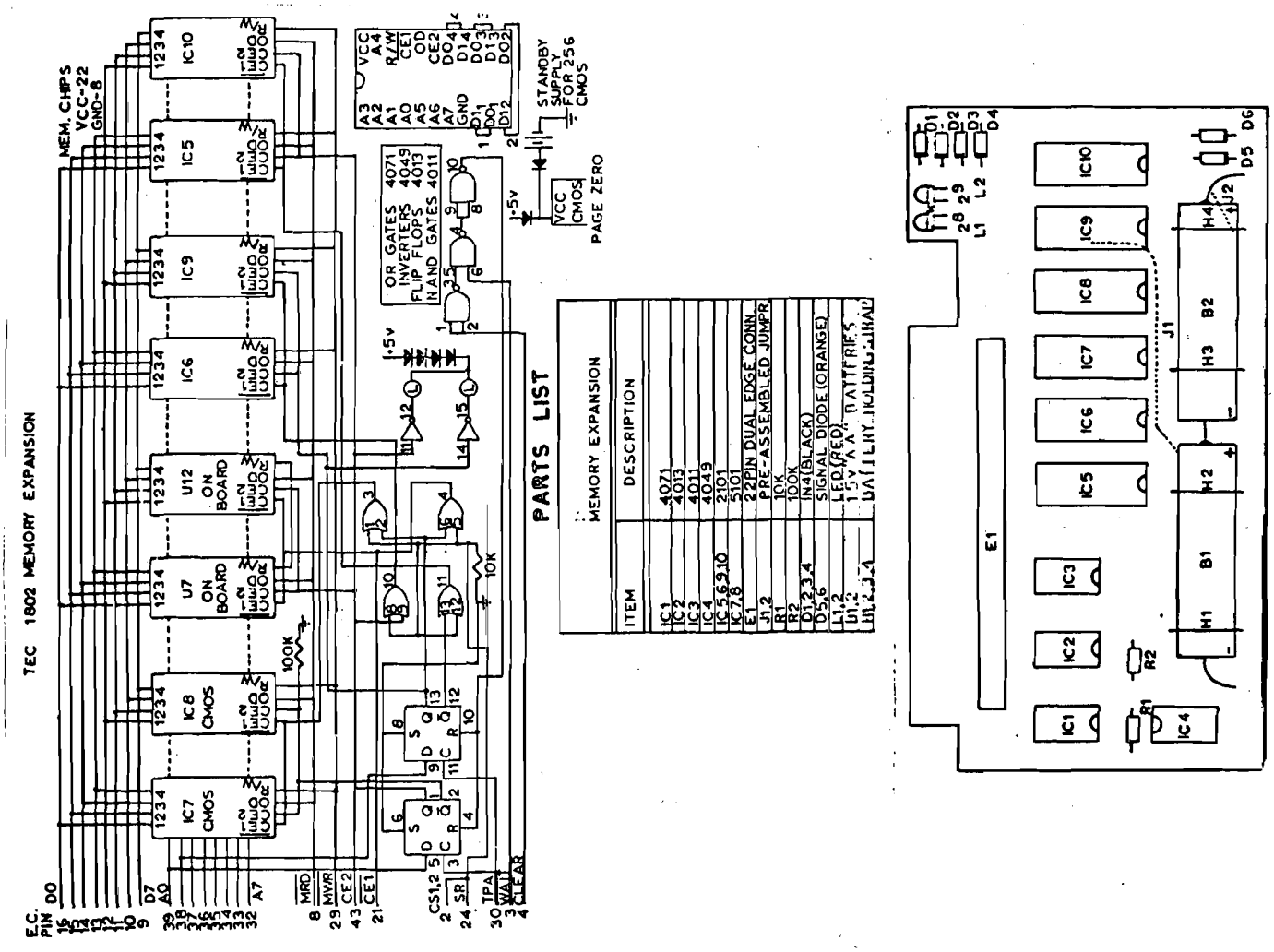
TEC-1802 Memory Expansion Board #1

E.M. Tekatch

This memory board provides an additional 768 bytes (3/4K) of memory. 256 bytes are implemented with CMOS RAM. When the main power supply is interrupted, the backup battery atuomatically maintains the CMOS RAM at approx. 10 ua. (good for about 1 year using 2 "AA" cells). The remaining 512 bytes is composed of NMOS RAM. The board draws about 80 ma. total.

Some minor changes are required on the 1802 main board, in order to provide the hardwired memory protect function for the expanded memory.

This approx. 3" X 6" PCB contains both a 22 pin dual edge connector, and a 22 pin dual edge connection. This allows the memory board to be connected directly to the CPU board, or to be plugged into one of the slots in a mother board. A kit containing the PCB and all parts except batteries is available for $59.32 (incl. taxes) from Tektron Equipment Corp., 263 Barton St., Unit 19, Stoney Creek, Ontario. Phone (416) 662-7820. Note that IC sockets are included with this kit, along with complete instructions.



Lottery Ticket Program

Nancy and Dougal Inkster

This program will check up to 128 tickets against an input, and will flash Q twice for every winning #.

Addr.	Inst.	Addr.	Inst.	Addr.	Inst.	Addr.	Inst.
00	F8	16	24	2C	12	42	D0
01	43	17	84	2D	3A	43	F8
02	AF	18	3A	2E	31	44	60
03	F8	19	11	2F	85	45	B6
04	4C	1A	94	30	F5	46	26
05	A2	1B	A4	31	12	47	96
06	A3	1C	3F	32	3A	48	3A
07	E2	1D	1C	33	3C	49	46
08	3F	1E	6C	34	7B	4A	30
09	08	1F	64	35	DF	4B	42
0A	6C	20	B5	36	7A		
0B	64	21	DF	37	DF		
0C	B4	22	3F	38	7B		
0D	22	23	22	39	DF		
0E	F4	24	6C	3A	7A		
0F	A4	25	64	3B	DF		
10	DF	26	A5	3C	24		
11	3F	27	83	3D	84		
12	11	28	A2	3E	3A		
13	6C	29	DF	3F	2A		
14	64	2A	95	40	30		
15	DF	2B	F5	41	1A		

Procedure: The three digits to be checked must be two bytes long, so NNN becomes 0NNN or NNN0. If the first three are also to be checked, count them as another ticket.

Entry Procedure:
1) Enter Program and Run.
2) Enter number of tickets and input.
3) Enter numbers on tickets and input.
4) Enter winning numbers.

Some machines bounce on input, so there is a software debounce, but if you enter the wrong number onto the bus when entering tickets, you have to reset because it counts the bytes. Also, be sure to enter high and low bytes in order. If you make a mistake when entering the high byte of a winner, you must still enter the low byte or they will be out of order.

Any # of winning numbers may be compared.

TWO SIMPLE SWITCH ADDITIONS FOR EASIER USE OF THE 1802

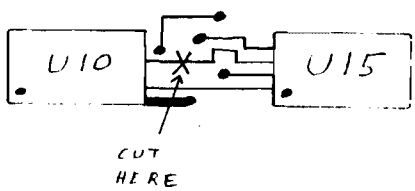
BY
BLAIR GERRISH

For anyone who would rather have a toggle switch than the small slide switch for the CONTROL/DATA switch, then the following change is very easy to make.

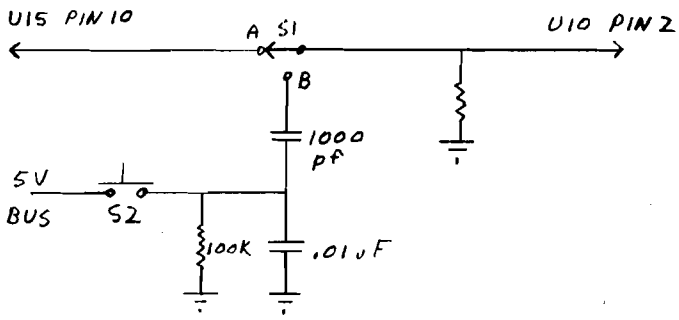
First, obtain a suitable size SPST switch and solder a wire from one pole of the switch to point Y on the PC board; this point is one which has one of the 3 wires from the keyboard soldered to it and it is marked on the PC board diagram. Then solder a wire from the other switch pole to the point on the PC board where the sixth wire of the group of 15 that connect the keyboard to the PC board is attached. This is the sixth wire from the left when looking at the top side of the PC board. The final step is to place the old C/D switch in the D position (or open position) and leave it there. When the new switch is closed, the keyboard will be in control mode; when open, it will be in data mode.

The second convenience switch will allow the user to step through memory 1 byte for each push of the switch, allowing for easier program verification or stepping to higher addresses of memory. This switch generates a DMA request to the CPU and may therefore have other uses to a particular user.

To install the circuit, first break the connection between U15 Pin 10 and U10 Pin 2. This can be found between the two IC's on the top side of the PC board as shown.



Install the following circuit.



When S1 is in position A, the keyboard will operate normally. When S1 is in position B, automatic DMA produced by every second keystroke will not have an affect on the CPU, instead pressing S2 will cause the DMA request to the CPU. If memory protection is on, then pressing S2 will cause the next sequencial byte to be displayed on the data LED's; if memory protection is not on, then the data presently in U17 will be loaded into memory. This may be useful if the user wants to store the same data in a number of sequential bytes since U17's status does not change while using S2 for DMA. This method does not use the available DMA-IN pad on the edge connector only because the owner of the computer wanted to reserve the edge connector for future use and wanted to be able to disable the keyboard DMA from U15 by installing S1.

NOTE: Use a good quality switch for S2 to help reduce switch bounce; otherwise, the debouncing filter shown may not be adequate.

TEC-1802 Editor Program

E.M. Tekatch

This program allows the running of a program from any address from 72 to 65K. Also the ability to examine, increment/decrement, modify, and load in any address from 72 to 65K.

00 (MA)	10 (MA)	20 (MA)	30 (MA)	40 (MA)	50 (MA)	60 (MA)
F8 66	B3 B4	6C	6C	2D	DE	21
AC	B5 B6	3A 4A	EE	FC 01	DF	91
F8 52	B7 B8	7A	3A 3A	3A 68	3F 52	3A 60
AD	B9 BA	EE	7A	2E 2E	F8 05	30 51
F8 67	BB BC	DD	DD	64	B1 21	00
AE	BD BF	6C	6C	30 2D	91	00
F8 1D	DF	BE	64	DD	3A 57	FC 01
AF	7B	DD	30 2D	6C	37 5B	3A 00
F8 00	EC	6C	FD 01	BE	F8 05	7A
B1 B2	DD	AE	3A 41	DD	B1	DD
		64	64	6C		6C
		C4	30	AE		64
		7B				30 6D
		EC				*Start of your program. (72 = MA)
		DD				

Operation:

1. Load carefully and check.

2. Run: A) Q led comes "ON" indicating an instruction request.
   You answer: 00 (press INPUT) editor mode.
   FF (press INPUT) run mode.

   B) 00 select - Q led goes "OFF" indicating a data or address request. Enter address 2 high-order digits (press INPUT), 2 low-order digits (press INPUT).

   C) Q led comes "ON" indicating instruction request.

   i) 00 (press INPUT) modifies data in next MA. Q led will go "OFF" when data is required. 00 must be repeated for each modification.

   ii) 01 (press INPUT) - step display forward one MA. Continuous pressing of "I" will increment MA ahead.

   iii) 02 (press INPUT) - step display back one MA.

Operation: (cont'd)

iv) 03 (press INPUT) - load data into next MA location; continuous loading with each "I". To discontinue, return with RESET/RUN to beginning.

00, 01, 02 mode may return to MA 00 (RESTART) by using FF for instruction input.

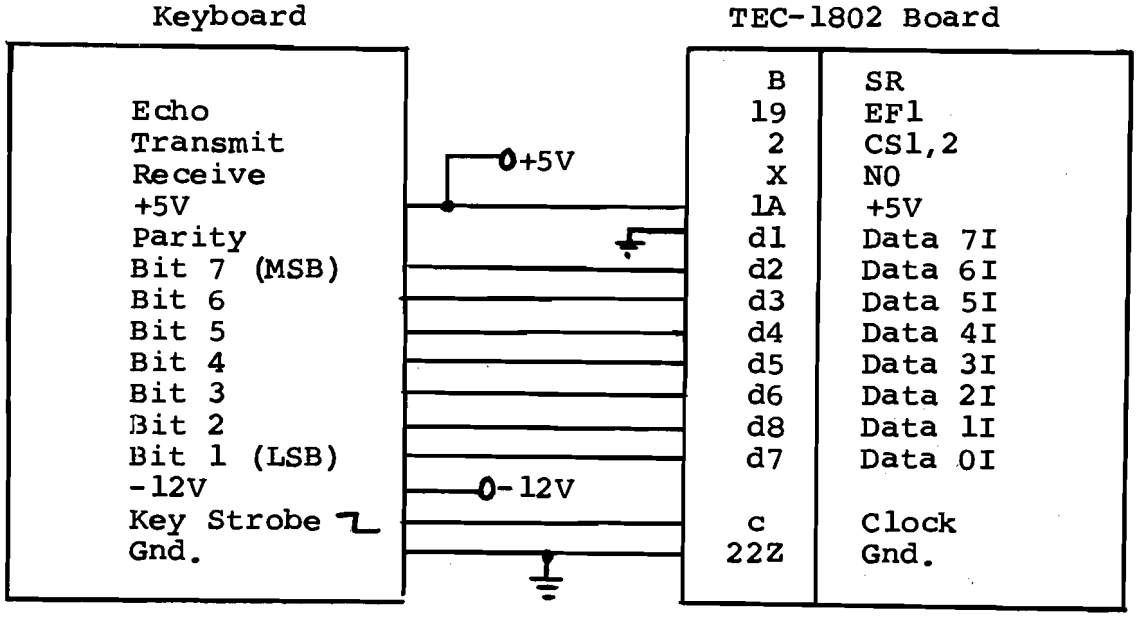
D) FF initial selection - Q "OFF"
Enter address of program you wish to run from 72 to 65K.

This program is used with the input pushbutton. In order to overcome C/D switching, SS and "I" can be switched.

Morse Code Program with ASCII Keyboard Input

Joe Kolodziej
Brian Fox

This program will accept data from an ASCII encoded keyboard, and transmit the typed character in Morse code. The keyboard is connected to the 1802 CPU board using an 1852 I/O chip. The ASCII data appears on 7 parallel input lines, and is signaled valid by sense line EF4 going to a logical " 1 " . Data inputs go into the 16 pin DIP socket above the 1852 Port. A drawing below shows the required hook-up from the keyboard to the Micro-processor Mother board. The keyboard used is Model KBD-5, from Southwest Technical Products ($50.00). For details on the abbreviations used in the drawing, see the TEC-1802 construction notes, page 12.



Note that this program will work with either the original 256 bytes of memory on the mother board, or with the memory expansion. In either case, the program loads starting at address zero.

```
                        .TITLE MORSE
                ;
                ;               MORSE CODE PROGRAM - ASCII KEYBOARD INPUT
                ;
0000 F8 00      BLANK:  LDI     #00             ; INIT. HIGH BYTE OF
0002 B3                 PHI     R3              ; ALL REGISTERS USED.
0003 B5                 PHI     R5
0004 B6                 PHI     R6
0005 B7                 PHI     R7
0006 B8                 PHI     R8
0007 BA                 PHI     R10
0008 BB                 PHI     R11
                ;
0009 F8 4C      SPEED:  LDI     WORK            ; WORK MEMORY LOCATION
000B AA                 PLO     R10
000C EA                 SEX     R10
000D 3F 0D      LOOP:   BN4     LOOP            ; WAIT FOR DATA FROM HEX
000F 6C                 INP4                    ; KEYBOARD AND ACCEPT DATA
0010 AB                 PLO     R11
0011 3C 11      KRBIN:  BR1     KRBIN           ; WAIT FOR,
0013 69                 INP1                    ; AND ACCEPT ASCII DATA
0014 64                 OUT4                    ; FROM KEYBOARD, AND DISPLAY.
0015 FC 20      MEMCON: ADI     #20             ; ADD CONSTANT, AND
0017 F9 C0              ORI     TABLE1          ; FORM INPUT ASCII DATA INTO
0019 A3                 PLO     R3              ; A MEMORY ADDRESS. STORE.
001A 2A                 DEC     R10             ; USE SAME WORK LOCS. LATER.
001B F8 37              LDI     DIT+1           ; START ADDR. OF DOT SUBR.
001D A5                 PLO     R5
001E F8 40              LDI     CNTR+1          ; START ADDR. OF COUNT SUBR.
0020 A6                 PLO     R6
0021 F8 09              LDI     #09             ; ACCUMULATOR + LINK SIZE
0023 A7                 PLO     R7
0024 03         DOIT:   LDN     R3              ; R3 POINTS TO DATA TABLE
0025 FE         TEST:   SHL
0026 D6                 SEP     R6              ; TEST FOR CODE BEGINNING
0027 3B 25              BNF     TEST
0029 FE         LETTER: SHL                     ; CHARACTER IF HERE
002A D6                 SEP     R6
002B 76                 SEQ                     ; START DOT/DASH
002C 3B 30              BNF     DOT             ; TEST FOR DOT OR DASH
002E D5                 SEP     R5              ; DASH
002F D5                 SEP     R5
0030 D5         DOT:    SEP     R5              ; DOT.
0031 7A                 REQ                     ; END OF DOT/DASH.
0032 D5                 SEP     R5
0033 08                 GLO     R8
0034 3B 29              BNF     LETTER          ; LETTER CONTINUATION LOOP
0036 D0         DIT:    SEP     R0              ; DOT SUBROUTINE DELAY
0037 8B                 GLO     R11
0038 B4                 PHI     R4
0039 24         COUNT:  DEC     R4
003A 94                 GHI     R4
003B 3A 39              BNZ     COUNT
003D 30 36              BR      DIT
003F D0         CNTR:   SEP     R0              ; COUNT DELAY SUBROUTINE
0040 A8                 PLO     R8
0041 27                 DEC     R7
0042 87                 GLO     R7
0043 32 48              BZ      RESET
0045 88                 GLO     R8
0046 30 3F              BR      CNTR            ; BRANCH IF 8 LOCATIONS NOT
0048 F8 03      RESET:  LDI     #0B-#5+1        ; CHECKED. OTHERWISE, RESET
004A A0                 PLO     R0              ; PC AND RETURN TO ACCEPT NEXT
004B D0                 SEP     R0              ; ASCII CHARACTER AT ADDRESS
                                                ; OF KRBIN.
004C    =   00 01       WORK:   .BLOCK  1       ; WORKING STORAGE
                ;
                ;               START OF ASCII TO MORSE CONVERSION TABLE
                ;
                                =   00 C0       .ORG    #C0
00C0                    TABLE1:
00C0 01                         .BYTE   #01,#00,#52,#31         ; SPACE , ! , " , #
00C1 00
00C2 52
00C3 31
00C4 89                         .BYTE   #89,#45,#28,#5E         ; $ , % , & , '
00C5 45
00C6 28
00C7 5E
00C8 6D                         .BYTE   #6D,#6D,#00,#2A         ; ( , ) , * , +
00C9 6D
00CA 00
00CB 2A
00CC 73                         .BYTE   #73,#61,#55,#32         ; , , - , . , /
00CD 61
00CE 55
00CF 32
00D0 3F                         .BYTE   #3F,#2F,#27,#23         ; 0 , 1 , 2 , 3
00D1 2F
00D2 27
00D3 23
00D4 21                         .BYTE   #21,#20,#30,#38         ; 4 , 5 , 6 , 7
00D5 20
00D6 30
00D7 38
00D8 3C                         .BYTE   #3C,#3E,#70,#6A         ; 8 , 9 , : , ;
00D9 3E
00DA 70
00DB 6A
00DC 00                         .BYTE   #00,#31,#00,#4C         ; < , = , > , ?
00DD 31
00DE 00
00DF 4C
00E0 35                         .BYTE   #35,#05,#18,#1A         ; @ , A , B , C
00E1 05
00E2 18
00E3 1A
```

```
MORSE                                   RC1802-V01      MORSE1.RCA      24-AUG-77  11:21:00

  0  00E4  0C                    .BYTE   #0C,#02,#12,#0E          ; D , E , F , G
  1  00E5  02
  2  00E6  12
  3  00E7  0E
  4  00E8  10                    .BYTE   #10,#04,#17,#0D          ; H , I , J , K
  5  00E9  04
  6  00EA  17
  7  00EB  0D
  8  00EC  14                    .BYTE   #14,#07,#06,#0F          ; L , M , N , O
  9  00ED  07
 10  00EE  06
 11  00EF  0F
 12  00F0  16                    .BYTE   #16,#1D,#0A,#08          ; P , Q , R , S
 13  00F1  1D
 14  00F2  0A
 15  00F3  08
 16  00F4  03                    .BYTE   #03,#09,#11,#0B          ; T , U , V , W
 17  00F5  09
 18  00F6  11
 19  00F7  0B
 20  00F8  19                    .BYTE   #19,#1B,#1C              ; X , Y , Z
 21  00F9  1B
 22  00FA  1C
 23                         ;
 24                         ;            NOTE... SOME LETTERS HAVE SPECIAL MEANINGS...
 25                         ;
 26                         ;    # -- BT - BREAK
 27                         ;    + -- AR - END OF MESSAGE
 28                         ;    & -- AS - WAIT
 29                         ;    % -- SK - END OF TRANSMISSION
 30                         ;    @ --    - COMMENCE
 31                         ;
 32                              .END
```

```
MORSE                                   RC1802-V01      MORSE1.RCA      24-AUG-77  11:21:00       PAGE  4
USER SYMBOL TABLE

BLANK  0000  CNTR   003F  COUNT  0039  DIT    0036  DOIT   0024  DOT    0030  KRBIN  0011  LETTER 0029
LOOP   0000  MEMCOM 0015  RESET  0048  SPEED  0009  TABLET 03C0  TEST   0025  WORK   014C

        PROGRAM SIZE = 0128
          0 ERRORS DETECTED
```

A Note on the Morse Code Program

Wayne Bowdish

The Morse Code Program was submitted for publication by Joe Kolodziej and Brian Fox. It was typed with assembly language code, machine code and comments. I'm sure the editor was extremely pleased with the article. The authors also supplied a patch sheet to allow the program to execute on systems with more than 256 bytes of memory (See TEC-1802 MEMORY EXPANSION BOARD #1). The editor decided to re-type the program and include these mods.

Since a preliminary version of the cross assembler (See A CROSS ASSEMBLER - WHAT'S IT MAD AT?) was available the editor decided to try it and see if it could assemble (a real Doubting Thomas). I would like to make some comments on the assembly.

First, I must apologize for the strange zeros. The listing was done on a well used Terminet 1200. Unfortunately, all the zero characters are broken and so small letter o's and diamonds have been substituted. Hopefully, this problem will be corrected on future examples.

A few comments on the layout of the listing are also in order. The top line of each page is a title line. I think this line is self explanatory. The body of the listing consists of 5 fields. The first number (decimal) is simply a line number. The second column contains a four digit (hex) address. The third field contains from one to three 2-digit (hex) numbers which are the data bytes. The fourth field may contain one or two error flags. In the example this field is blank since there are no errors. Finally, the assembly language program source lines are listed exactly as received by the assembler. A source line consists of labels, opcodes, operands and comments. These fields may or may not be present.

There are several conventions which should be noted. First, all labels in the label field are followed by a colon. All comments begin with a semi-colon. Opcodes and operands (if present) are separated by spaces or tabs. Another convention concerns hex constants. The character # is used to signify a hex constant (decimal, octal, and ASCII constants are also allowed). Registers may be specified by constants or the special symbols R0,R1, ..., R14,R15 may be used.

Finally, all pseudo-ops begin with a period "." . Two pseudo-ops were used in the Morse Code Program, .ORG and .BYTE. The .ORG pseudo-op simply sets the program counter to some value, #00C0 in the example. This is useful for locating blocks of code or data at specific addresses. The .BYTE pseudo-op places a constant in a memory byte.

In summary, I'm happy to see that somebody has found a use for the assembler. Possibly future newsletters will contain more samples of assembler output which will demonstrate more of the assembler's capability.