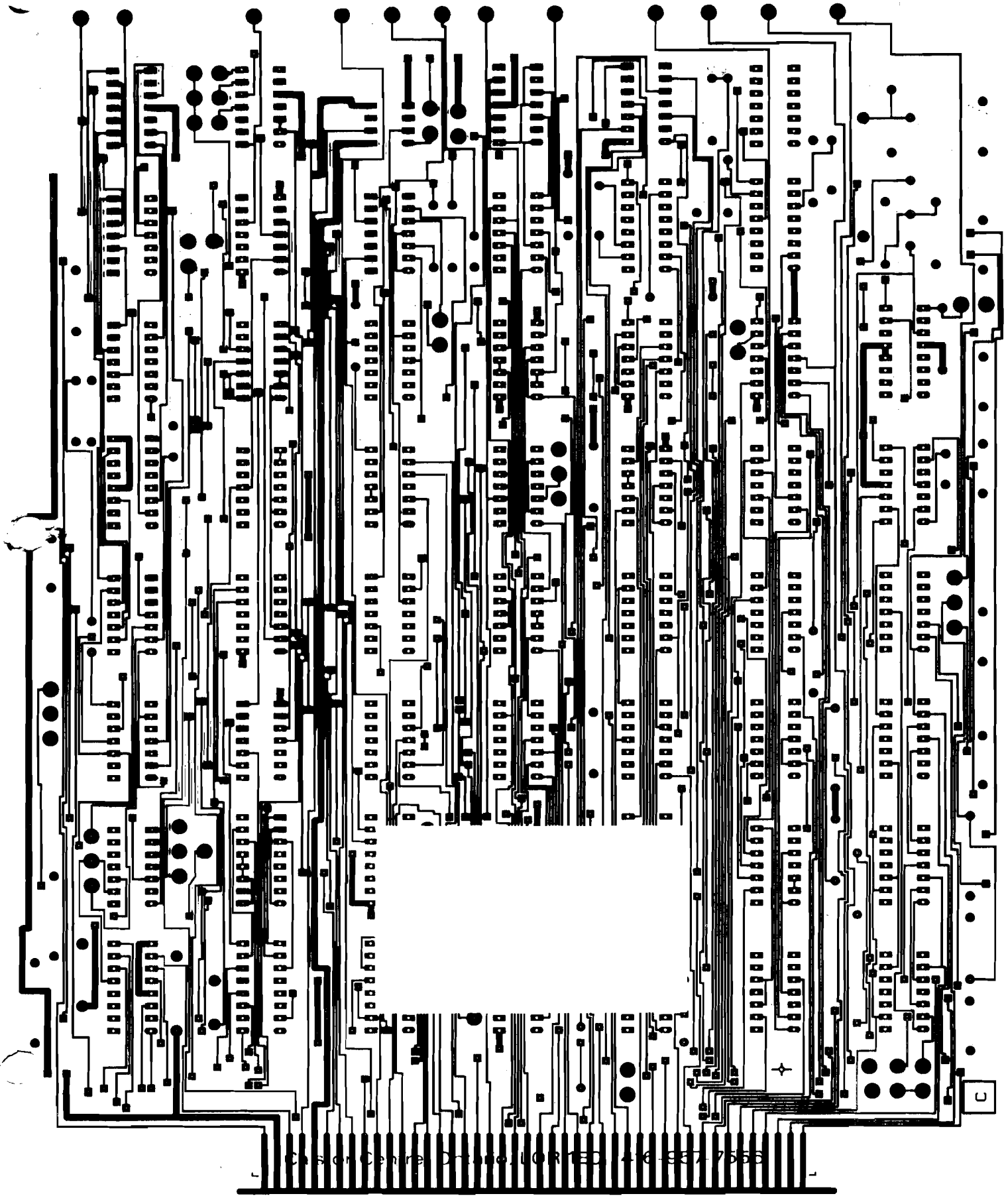
TEKTRON EQUIPMENT CORPORATION
Caistor Centre, Ontario, LOR 1EO 416-957-7556
TEC-NEWS LETTER ISSUE #1
MAY 77



# IN STOCK ELECTRONIC COMPONENTS

12% Federal & 7% Provincial Tax Extra
F.O.B. our plant - Minimum Order $5.00

Prices Effective:
May 1, 1977.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2101 | $3.55 | 3055 with lugs | .95 | 7411 | - | .45 | 7474 | - | .42 |
| 2102 | 2.10 | | | 7420 | - | .19 | 7475 | - | .55 |
| 1702 New | 13.50 | 7400 | - .18 | 7432 | - | .38 | 7486 | - | .33 |
| 1702 Used | 9.50 | 7402 | - .18 | 7447 | - | .89 | 7490 | - | .55 |
| 309K | - 2.95 | 7404 | - .22 | 7451 | - | .25 | 7493 | - | .55 |
| LM3900 | - .79 | 7408 | - .25 | 7473 | - | .32 | 74123 | - | .80 |
| | | 7410 | - .19 | 555 | - | .55 | 74157(8123) | | .75 |

| | | | |
|---|---|---|---|
| Man. 1 .3" Display | $3.00 ea. | assorted lug terminal strips | 1.00/25 |
| 22 pin dual edge conn. P.C. | 2.95 | .01 mf ceramic capacitor | .06ea. 4.50/100 |
| 60 conductor flat ribbon cable | 1.00/5' | .01 mf mylar capacitor | .07ea. 5.00/100 |
| 14 pin socket | .25 | .1 mf " " | .09ea. 7.00/100 |
| 16 pin socket | .35 | 2200 mf 16v | .46 |
| 22 pin socket | .50 | 1000 mf 25v | .32 |
| 24 pin socket | .60 | IN4005 1A diode | .09ea. 7.00/100 |
| 2 wire line cord 6' | .30 | IN914 signal diode | .05ea. 4.00/100 |
| electronic eyelets 1/16" PCB | 1.50/100 | 1/4w5% resistors(all sizes) | .04ea. 3.00/100 |
| OD .047" ID .032" FLG. .080 | | EL. SK10 socket | 23.25 |
| mixed random cut PCB | $8.00/10LBS. | education price | 20.50 |

| | | |
|---|---|---|
| PESC - 1 lb. | Photo Engravers Scrub Cleaner 1 lb. | 1.62 |
| PESC - 10 lbs. | " " " " 10 lbs. | 15.00 |
| NCC - 1G | Neutral Copper Cleaner Solution 1 gal. | 9.15 |
| CITP - 1PT. | Cold Immersion Tin Plating 1 pt. | 4.55 |
| CITP - 1G | (shelf life 3 to 6 months) 1 gal. | 18.00 |
| Riston Laminating (min. 24 sq. ft.) 1/s | | 2.45/sq.ft. |

| | | FST & PST |
|---|---|---|
| TEC - 1802 Microprocessor Kit (HIGH speed chip) | $106.00 | $127.00 |
| Mother Board. - 5, 22 pin dual edge conn. 4" x 6" | 19.00 | 22.75 |
| I.C. socket kit for TEC - 1802 | 7.95 | 9.50 |
| Input parallel port 1852 & socket | | 16.00 |
| 5V .4A Regulated Power supply (good for 1A with larger heat sink) | 10.00 | 12.00 |
| CMOS Memory brd. 256 x 8 CMOS 512 NMOS. | 52.00 | 62.30 |
| 1802 D | 34.00 | 40.74 |
| Low Power LEDS | 5/1.00 | 5/1.20 |
| Minature Toggle Switch | 1.15 | 1.36 |
| TEC-1802 PCB | | 32.00 |
| Reconditioned Digital Equipment Corp. Computer Logic Lab | | 150.00 |

Comming PCB

- 4K RAM. static NMOS
- super arithmetic unit
- A/D converter (8 channel)
- UART casset/teletype interface
- heavy acrylic case

## WHAT DO I DO NOW ???

No doubt many of us are asking this question: Now that I have a micro-processor, what can I do with it?

Most people will poke around with a few programs, maybe build a bit of hardware to perform some specific task, and perhaps write a few interesting subroutines. By itself, this may not add up to much. But what if, somehow, you could find out what other people have been doing with their micros? You could perhaps use their ideas in combination with your own to produce a bigger, better and more interesting application for your own micro-processor.

What we intend to do is provide a forum for people to present their ideas and applications for the 1802 micro-processor kit. The people who have produced the kit and the course will be subsidizing the first few issues. The existence of this newsletter past that point will dependent primarily upon your interest and participation. Remember; this newsletter will be only as useful as the articles it presents, and these articles must be supplied by you.

Possible ideas for presentation can be software or hardware, or both, and they needn't be restricted to 1802 micro-processors. Most software and hardware can be easily converted to any machine you choose. 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 some one else can help you out with some practical suggestions, or perhaps they have already started work along the same lines.

We have a list of ideas we would like to hear more about, and also some projects of our own which you might find interesting.

Software

* - a UART function (Universal Asynchronous Receiver/Transmitter) implemented in software, and using the Q-line for output and an I/O Flag line for input. This could be used with a Serial Terminal, a cassette storage unit, or a telephone modem.
  - a set of routines to perform addition, subtraction, multiplication, and division on 16 bit Integer numbers.
  - another set to perform arithmetic on floating point numbers.
  - some scientific function subroutines ($e^x$, log, ln, sine, etc.) for either of the 2 numbering systems mentioned above.
  - a real-time clock program.
  - a Software Interrupt driver to work in conjunction with some simple vectored interrupt hardware.
  - a multi-programming executive.
  - a cassette storage executive.
  - an editor routine, and a simple assembler.
* - a cross-assembler, for use on another, large computer, to produce assembled code for the 1802.
  - a BASIC Interpreter program.
  - plus uncounted individual applications programs.

Hardware

* - a case, power supply, and mother board for the existing micro-processor kit.

* - more memory for the 1802.
  - some elementary vectored interrupt hardware.
  - a cassette storage system for programs, based on an inexpensive cassette tape recorder.
* - a hardware UART function for serial communications.
  - a simple D/A convertor for outputting to scopes and recorders, and for music synthesis.
  - an A/D convertor, to use in a micro-processor based multi-meter, for example.
  - some hardware to use your micro-processor as a Logic Analyzer.
  - a calculator interface, to implement math functions.
  - and many more.

The list of projects is at least as long as the list of imaginative 1802 kit owners (I think this includes most of the people on our mailing list!). The asterisks above indicate projects which people are presently working on. You will find some of them described in this and future issues of this newsletter.

We also intend to have an informal letters-to-the-Editor (me) column, and a column for items for sale or swap, or wanted. There may even be some advertising from suppliers of parts and equipment (besides the supplier of the 1802 kit). It may also be necessary to charge a small subscription fee to this newsletter, to defray costs.

But remember, all of this activity is for one purpose only: the exchange of useful information. Even if you don't consider your efforts particularly newsworthy, there are bound to be several people amongst the nearly 300 current kit owners who are working on something similar to what you have done. Also, if you have any thoughts on the format of this newsletter, or if you think this forum is a particularly good (or bad) idea, please let us know.

Tom Crawford
Editor

In order that this newsletter will improve and grow we need your support. Write or call me with articles for publication, items for sale or general information.

After numerous requests I have been considering an advanced coarse in programming and application for the TEC-1802. Let me know what you think.

Eugene M. Tekatch
Publisher
Tektron Equipment Corp.
Caistor Centre, Ont.
L0R 1E0
Phone: 1-957-7556

Music and Micro's

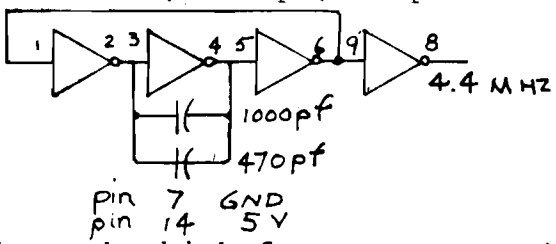
This was a successful theme for Anthony Tekatch in two science fairs. The following is his program with a few hardware changes to obtain the speed and output drive.

| MA | INSTRUCTION | DESCRIPTION |
|---|---|---|
| 00 | F8 10 A4 | initialization |
| 03 | F8 45 A2 | . |
| 06 | F8 43 A5 | . |
| 09 | F8 FE A8 | . |
| 0C | F8 41 A7 | . |
| 0F | 7A | Reset Q |
| 10 | 3F 19 | input data for |
| 12 | E5 | desired beat |
| 13 | 6C | . |
| 14 | B6 | . |
| 15 | 64 | . |
| 16 | 25 | . |
| 17 | 37 17 | . |
| 19 | 02 57 | test for FF (rest) |
| 1B | E7 | and execute a pause |
| 1C | 88 | if found |
| 1D | F5 | . |
| 1E | 3B 28 | . |
| 20 | 05 B9 | . |
| 22 | 29 | . |
| 23 | 99 | . |
| 24 | 3A 22 | . |
| 26 | 30 37 | . |
| 28 | 02 A3 | delay loop |
| 2A | 23 | . |
| 2B | 26 | . |
| 2C | 96 | . |
| 2D | 32 37 | branch out |
| 2F | 83 | . |
| 30 | 3A 2A | . |
| 32 | 31 0F | . |
| 34 | 7B | set Q |
| 35 | 30 19 | |
| 37 | 12 | |
| 38 | 05 B6 | select next note |
| 3A | 24 | |
| 3B | 84 | |
| 3C | 32 00 | end song, repeat |
| 3E | 30 0F | |
| 45 | song | |

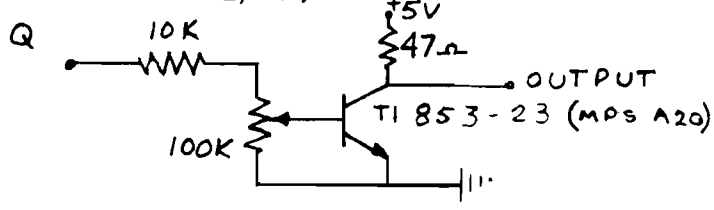
NOTES

| High | | | | | | | | | | | Low | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| F | E | D# | D | C# | C | B | A# | A | G# | G | F# | F | E | D# | D | C# | C | B |
| 18 | 1A | 1C | 1D | 1F | 21 | 24 | 26 | 28 | 2B | 2E | 31 | 34 | 38 | 3A | 3E | 42 | 26 | 4A |

High frequency oscillator
- requires 7404, 1000pf, 470pf




Output drive
- requires 10K-1/4w, 100K pot, 47Ω-1/4w, NPN driver




Operation: The high frequency operation requires the D chip. Enter the program as shown with your song starting at location 45. Put in the appropriate hexidecimal number for the note and insert FF for a rest. Enter in MA 01 the total number of beats in the song for automatic repeat. More than one song could be stored by changing the starting address in MA 04 to correspond to some other area of memory. When the program is put in the run mode enter the beat and the music starts on the release of I.

The local Hamilton club has been active in applying the TEC-1802 to their specific area. The MORSE CODE program has been upgraded by Joe Kolodziej (Picker Xray) and Brian Fox (Radio Shack).

With many active people working on this we should soon have a teletype transmitter and CRT receiver for Morse Code.

Code Program by Brian Fox VE3EBF

| START HERE | CONTINUE HERE |
|---|---|
| 00 F8 44 | 22 86 |
| 02 A5 | 23 32 04 |
| 03 E5 | 25 26 |
| 04 F8 18 | 26 87 |
| 06 BB | 27 7B |
| 07 2B | 28 F8 10 |
| 08 9B | 2A B8 |
| 09 3A 06 | 2B F8 08 |
| 0B C4 | 2D B9 |
| 0C C4 | 2E F8 08 |
| 0D F8 08 | 30 BA |
| 0F A6 | 31 3B 37 |
| 10 F0 | 33 28 |
| 11 64 | 34 98 |
| 12 C4 | 35 3A 33 |
| 13 C4 | 37 29 |
| 14 7A | 38 99 |
| 15 C4 | 39 3A 37 |
| 16 C4 | 3B 7A |
| 17 FE | 3C 2A |
| 18 A7 | 3D 9A |
| 19 86 | 3E 3A 3C |
| 1A 32 04 | 40 87 |
| 1C 26 | 41 FE |
| 1D 87 | 42 30 21 |
| 1E 3B 12 | |
| 20 FE | |
| 21 A7 | |

ALPHABET TABLE

| | |
|---|---|
| A-05 | 0-3F |
| B-18 | 1-2F |
| C-1A | 2-27 |
| D-0C | 3-23 |
| E-02 | 4-21 |
| F-12 | 5-20 |
| G-0E | 6-30 |
| H-10 | 7-38 |
| I-04 | 8-3C |
| J-17 | 9-3E |
| K-0D | |
| L-14 | SPACE 01 |
| M-07 | BT 31 |
| N-06 | PERIOD 51 |
| O-0F | |
| P-16 | |
| Q-1D | |
| R-0A | |
| S-08 | |
| T-03 | |
| U-09 | |
| V-11 | |
| W-0B | |
| X-19 | |
| Y-1B | |
| Z-1C | |

Start programming your code massage at address 44, by using the alphabet table. Use word space code 01 between each word.

To change to a slower speed change addr. 05 to 30
29 to 20
2C to 10
2F to 10

To change to a faster speed change addr. 05 to 0C
29 to 08
2C to 04
2F to 04

More characters can be generated as long as the total count does not exceed 7. Place A 1 in front of the letter making a total count of 8. See example below .....Dots 0 Dashes 1

BT -...- 0001 0001 Add A 1 in front 0011 0001 Thus 31 HEX

PERIOD .-.-.- 00010101 Add A 1 in front 0101 0101 Thus 51 HEX

# Light Pattern Display Program

Blair Gerrish
Dofasco Engineering Dept.

The following program can be used to display a pattern of bytes on the data bus LED display. The storage area for the bytes to be displayed starts at location 02 and runs to 11. The program will display the bytes sequentially starting with location 02 and delaying after each byte until it either encounters a 51 stored in a byte or until it goes through the entire storage area; at this point it will start over again. Pushing the I button will cause the program to stop until I is released. The delay time constant is stored in location 12 by the user. If the user wishes to change the stop constant which is 51 in this program, then change location 1F to the value that is desired. When the program finds a byte which matches the stop constant, then the display will start over. The data buffer and delay time are stored at the beginning of the program to make them easier to change; be careful not to destroy the contents of locations 00 or 01 when changing the data or delay time.

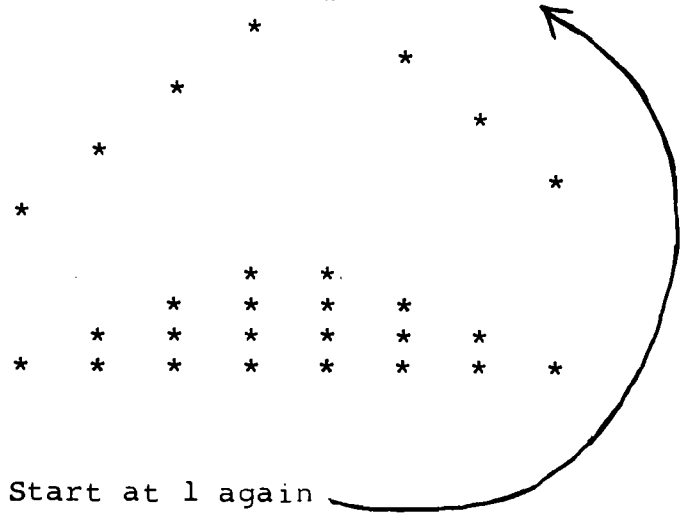
| ADDRESS | INST. | ASSEMBLY LANGUAGE | | | COMMENTS |
|---|---|---|---|---|---|
| 00 | 30 | | BR | ENDTAB | branch over data area to start |
| 01 | 13 | | | | program |
| 02 | 08 | BEGTAB | BSS | 10 | locations 02 to 11 are for |
| 03 | 10 | | | | the data that is to be |
| 04 | 04 | | | | displayed, the data given |
| 05 | 20 | | | | is a suggested starting pattern |
| 06 | 02 | | | | |
| 07 | 40 | | | | |
| 08 | 01 | | | | |
| 09 | 80 | | | | |
| 0A | 00 | | | | |
| 0B | 18 | | | | |
| 0C | 3C | | | | |
| 0D | 7E | | | | |
| 0E | FF | | | | |
| 0F | 00 | | | | |
| 10 | 51 | | | | display will start again upon |
| 11 | 00 | | | | reaching this point. |
| 12 | 20 | DELTIM | BSS | 1 | delay time count, try 20 to start |
| 13 | F8 | ENDTAB | LDI | BEGTAB | set up pointer into data buffer |
| 14 | 02 | | | | |
| 15 | A3 | | PLO | 3 | in R3.0 |
| 16 | F8 | | LDI | DELTIM | set up pointer to delay time |
| 17 | 12 | | | | |
| 18 | A4 | | PLO | 4 | in R4.0 |
| 19 | F8 | | LDI | 10 | put maximum size of data |
| 1A | 10 | | | | area in |
| 1B | A6 | | PLO | 6 | R6.0 |

... continued

| ADDRESS | INST. | ASSEMBLY LANGUAGE | | | COMMENTS |
|---|---|---|---|---|---|
| 1C | E3 | | SEX | 3 | set x to 3 for output |
| 1D | 03 | NEXT | LDN | 3 | load via R3 to get data |
| 1E | FD | | SDI | 51 | subtract 51 |
| 1F | 51 | | | | if result=0 start again |
| 20 | 32 | | BZ | ENDTAB | by branching to location 13 |
| 21 | 13 | | | | if result≠0 then |
| 22 | 64 | | OUT4 | | display the data. Increment R3 |
| 23 | 04 | | LDN | 4 | load via R4 to get delay time |
| 24 | B5 | | PHI | 5 | put delay time in R5.1 |
| 25 | 25 | DECAGN | DEC | 5 | and count it down |
| 26 | 95 | | GHI | 5 | get R5.1 |
| 27 | 3A | | BNZ | DECAGN | if R5.1≠0 go and decrement again |
| 28 | 25 | | | | otherwise continue |
| 29 | 37 | HERE | B4 | HERE | if EF4=1 (I button being pressed) |
| 2A | 29 | | | | then stay here |
| 2B | 26 | | DEC | 6 | decrement register 6 and check |
| 2C | 86 | | GLO | 6 | it for 0, if 0 then entire |
| 2D | 32 | | BZ | ENDTAB | data area has been displayed |
| 2E | 13 | | | | so start again |
| 2F | 30 | | BR | NEXT | otherwise, get next location |
| 30 | 1D | | | | to display. |

This program should repeat the following pattern:

| LOOP | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| 1 | | | | | * | | | |
| 2 | | | | * | | | | |
| 3 | | | | | | * | | |
| 4 | | | * | | | | | |
| 5 | | | | | | | * | |
| 6 | | * | | | | | | |
| 7 | | | | | | | | * |
| 8 | * | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | * | * | | | |
| 11 | | | * | * | * | * | | |
| 12 | | * | * | * | * | * | * | |
| 13 | * | * | * | * | * | * | * | * |
| 14 | | | | | | | | |



# A SUBROUTINE FOR PSEUDO-RANDOM NUMBER GENERATION

By Tom Crawford

The routine shown in Listing 1 has been converted from Motorola 6800 Assembly language, to run on an RCA 1802 micro-processor. The original article, entitled "Randomize Your Programming", can be found in the September, 1976 issue of BYTE Magazine, Page 36-39.

This version will provide a series of 217 pseudo-random numbers before repeating itself. Note this is not a uniform distribution, over the 0 to 255 range.

To use the subroutine, simply set the contents of Register 5 to point at the memory location you would like the data returned in. Then call the subroutine by whatever method is appropriate (I have used PC-pointer switching in Listing 1, using R4 for subroutine PC, and R3 for main line PC). Upon return from the subroutine, the data location will contain the next pseudo-random number in the series.

I have hand-assembled the routine, beginning arbitrarily at memory location C1 (not C0; that's the return point).

Listing 1

| Address | Inst. | Assembly Language | | Comments |
|---|---|---|---|---|
| C0 | D3 | RETURN | SEP R3 | |
| C1 | 05 | ENTRY1 | LDN R5 | Get data into D |
| C2 | 3A C7 | | BNZ NOTO | branch if D ≠ 0 |
| C4 | FC 01 | | ADI #01 | add 1 to D if D=0 |
| C6 | 55 | | STR R5 | |
| C7 | FA 84 | NOTO | ANI #84 | mask out feedback bits |
| C9 | 32 D9 | | BZ SKIP | if no feedback bits then exit. |
| CB | FE | SHIFT | SHLC | else loop to find lst set bit. |
| CC | 3B CB | | BNF SHIFT | |
| CE | 3A D9 | | BNZ SKIP | |
| D0 | 05 | | LDN R5 | |
| D1 | FE | | SHL | shift previous data left, |
| D2 | FC 01 | | ADI #01 | add 1, |
| D4 | FA FF | XIT | ANI #FF | mask off unused bits |
| D6 | 55 | | STR R5 | and save the result as new data. |
| D7 | 30 C0 | | BR RETURN | restore R4 to start before returning. |
| D9 | 05 | SKIP | LDN R5 | |
| DA | FE | | SHLC | shift previous data left. |
| DB | 30 D4 | | BR XIT | |

For those people who would like a smaller set of pseudo-random numbers, or uniform distribution of pseudo-random numbers, Table A gives values for some different ranges. To use these, simply substitute the Mask 1 byte for the mask byte found at address C8 in Listing 1, and substitute Mask 2 for the mask byte at address D5.

Table A

| Stages | Period | Mask1 | Mask2 | Range |
|---|---|---|---|---|
| 2 | 3 | 03 | 03 | 1-3 |
| 3 | 7 | 06 | 07 | 1-7 |
| 4 | 15 | 0C | 0F | 1-15 |
| 5 | 31 | 14 | 1F | 1-31 |
| 6 | 63 | 30 | 3F | 1-63 |
| 7 | 127 | 60 | 7F | 1-127 |
| 8 | 217 | 84 | FF | 1-255 |

A test routine is provided in Listing 2. I have arbitrarily assembled this routine to start at location 00, and it will run with Register 3 as the Program Counter. Each time the "I" pushbutton is pressed, the next pseudo-random number in the series will appear on the data leds.

Listing 2

```
Address   Inst.     Assembly Language      Comments

  00      F8 C1          LDI RANDOM        Set up the subroutine
                                           pointer.
  02      A4             PLO R4
  03      F8 14          LDI DATA          set up the data pointer
  05      A5             PLO R5
  06      E5             SEX R5            X points to the data
                                           also
  07      F8 0B          LDI START         set up the PC
  09      A3             PLO R3
  0A      D3             SEP R3            set Register 3 to PC
  0B      3F 0B  START   BN4 START         loop until I is pushed
  0D      D4             SEP R4            call subroutine.
  0E      64             OUT4              send random number to
                                           data leds
  0F      25             DEC R5            restore X Register after
                                           output
  10      37 10  LOOP    B4  LOOP          loop until I is released
  12      30 0B          BR  START

  14              DATA   BSS 1
```