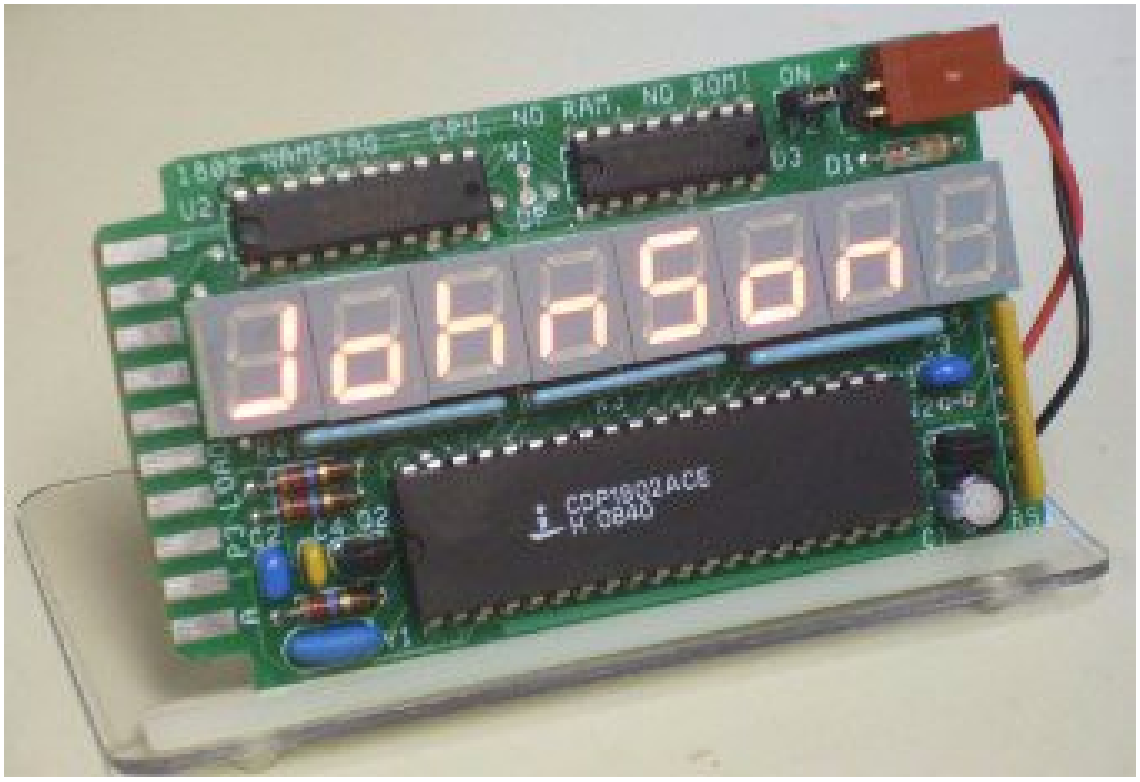
An 1802 Name Tag

A TMSI **ElectroniKit** (tm)

http://www.sunrise-ev.com/membershipcard.htm#nametag



My latest little project is this 1802 Name Tag. It shows how much you can do with an 1802 and an absolute minimum of resources! It's the size of a credit card, and draws about 10ma -- low enough to run for many days on three AAA cells or a single 3.6v lithium cell.

The 1802 scans a multiplexed 8-digit 7-segment LED display to produce four 8-character message "frames", displaying them sequentially about one every second. Every segment in each message is programmable, so you can spell out text as well as numbers.

Astute observers will note that there is no RAM or ROM! Yet, it is running a program and scanning the display. It even has a 'front panel' interface for loading messages, which since it is an 1802, needs nothing but switches and LEDs.

This is a great "quickie" project! The kit comes with all the on-board parts. You'll need to supply a battery (and battery holder) to power it. You also need nine SPDT center-off toggle switches, nine 3.3k resistors, an LED, and an edge connector to build the Programming Panel shown on the schematic. (I haven't had time to make a PC board for the programming panel yet). Ready? Then let the fun begin!

If you bought a kit, find each part, and check it off in the space provided. If you bought a bare board, this is your "shopping list". Leave the IC chips in their protective packaging, as they can be damaged by static electricity.

by Lee Hart, 814 8th Ave N, Sartell MN 56377, leeahart@earthlink.net

Parts List

Quantity		Identifier	Description
( )	1	C1	Capacitor, 33uF 25v electrolytic (jameco.com 2125587)
( )	2	C2, C3	Capacitor, 0.1uF 50v ceramic (jameco.com 544868)
( )	1	C4	Capacitor, 470pF 50v ceramic (digikey.com 399-9740-ND)
( )	1	D1	Diode, 30v 1a Schottky 1N5818 (jameco.com 177957)
( )	8	LED1-8	LED, 7seg 0.3" common anode (jameco.com 2188895)
( )	1	JP2	shorting jumper for P2 on-off switch (jameco 22024)
( )	1	P1	2-pin header, right angle 0.1" center (jameco.com 2120305)
( )	1	P2	2-pin header, straight 0.1" centers (jameco.com 2144884)
( )	2	Q1, Q2	Transistor, 2N7000 N-chan MOSFET (jameco.com 178669)
( )	3	R1, R6, R7	Resistor 10meg, brown-black-blue-gold (jameco.com 691817)
( )	3	R2, R3, R4	Resistor network, 4 x 10k isolated, SIP8 (jameco.com 856909)
( )	1	R5	Resistor network, 4 x 2.2k isolated, SIP8 (jameco.com 1971327)
( )	1	U1	CDP1802 microprocessor (eBay, or from me:-)
( )	1	U2	74HC241 octal buffer (digikey.com 296-8271-5-ND)
( )	1	U3	74HC138 decoder (jameco.com 45330)
( )	1	Y1	1.8 MHz ceramic resonator (eBay, or from me :-)
( )	1	PCB	"1802 NAMETAG" printed circuit board (from me :-)

Parts for the Programming Panel (not supplied)

( )	9	R8-R16	Resistor 3.3k, orange-orange-red-gold (jameco.com 690988)
( )	1	D2	LED, red T1-3/4 (jameco.com 2125309)
( )	9	S0-S8	Toggle switch, SPDT center-off (jameco.com 317244)
( )	1	P3	edge connector, 20-pin 0.156"ctr (digikey.com 151-1190-ND)

Assembly

If you've built any electronic projects before, assembly is pretty straightforward. You'll need a soldering gun or iron, rosin-core electronics solder (please; not acid-core plumbing solder!), and wire cutters.

In the grand old Heathkit style, step-by-step instructions start on the next page. I like to put the lowest height parts in first, and work up to the tallest ones last. This means resistors first, then the diode, ICs, capacitors, transistors, and finally the LEDs. Hints:

- Do the steps in order, then mark the box (X) so you know you did it.
- All parts go on the **front** side of the board, marked "1802 NAMETAG".
- Insert each part, then bend a couple of its pins or wire leads slightly to hold it in place.
- Then turn the board over, and solder all the part's connections.
- Finally, cut off any excess wire length.

Resistors are "dogbone" shaped tubes with color rings to indicate their resistance, and a wire on each end. Bend the wires at a right angle, and insert them into the holes on the board.

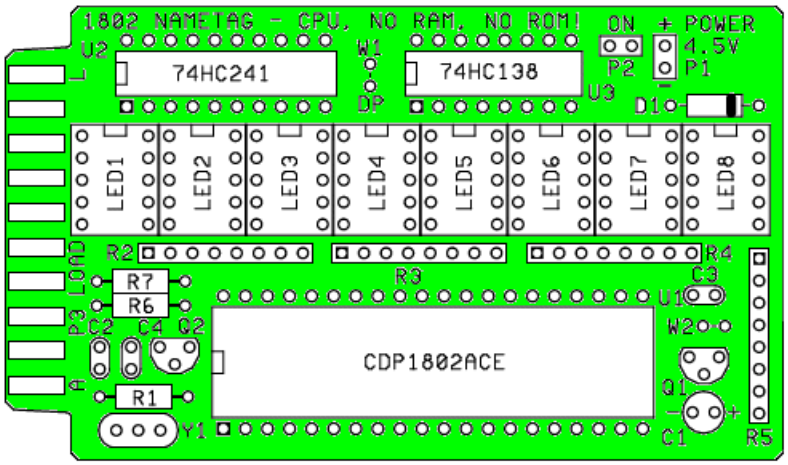
( ) R1: 10meg (tan body, with brown-black-blue-gold rings).
( ) R6: 10meg " "
( ) R7: 10meg " "

Diode D1 is a black tube with a wire on each end. Bend the wires as you did for the resistors. Diodes must be installed the right way! The banded end must match the label on the board.

( ) Diode D1 (marked 1N5818).

Resistor networks look like a stick with 8 pins coming out the side. One end has a dot or stripe; it goes in the hole with the square pad, toward the top or left side of the board.

( ) R2: 10k x 4 isolated (black body, marked L83S103).
( ) R3: 10k x 4 " " " "
( ) R4: 10k x 4 " " " "
( ) R5: 2.2k x 4 isolated (black body, marked 10B222G).



ICs (Integrated Circuits) look like black "caterpillars". Bend the pins inward slightly, and insert it onto the board. ICs must face the right way! The end with the notch must match the notch on the board. The text on the IC should be right side up and match the rest of the text on the board. I didn't provide IC sockets, but you can add them if desired.

( ) U3: 74HC138 (marked "SN74HC138N").
( ) U2: 74HC241 (marked "SN74HC241N").
( ) U1: 1802 (marked "CDP1802ACE").

Capacitors have two wires coming out the same side.

( ) C1: 33uF (black body, marked "33uF 25v"). The wire next to the white stripe and big "-" sign goes in the hole with the "-" sign.
( ) C2: 0.1uF (blue body, marked "104M").
( ) C3: 0.1uF " " "
( ) C4: 470pF (yellow body, marked "471").

The **Ceramic Resonator** looks like a capacitor, but it has 3 leads.

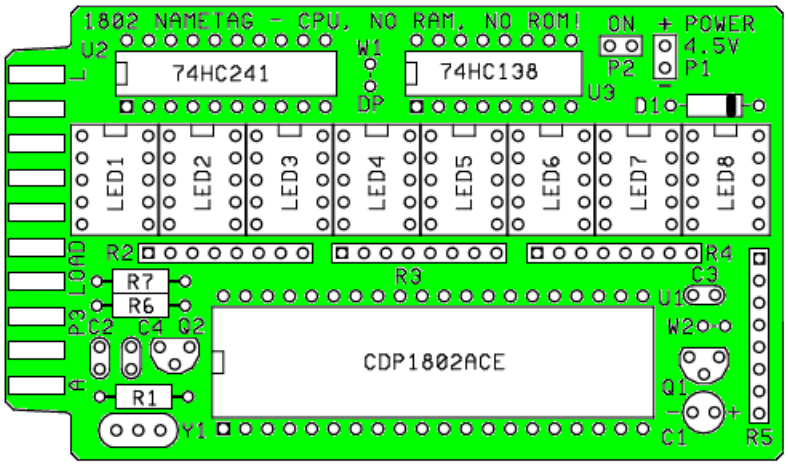
( ) Y1: 1.8 MHz ceramic resonator (blue, marked "1.80Y").

Transistors have a black plastic case with three wires coming out the same side. One side of the case is flat. This side **must** match the flat side printed on the board. Bend the center wire slightly away from the flat side, and push the transistor close to the board.

( ) Q1: 2N7000
( ) Q2: 2N7000

Connectors are installed like the other parts.

( ) P1: A right-angle 2-pin header. The pins point to the left in the illustration below.
( ) P2: A straight 2-pin header.



The **LEDs** must be installed with the decimal point in the lower right corner (with the board positioned as shown below). The labeling will be on the right side of the LEDs.

( ) LED1: (a 10-pin white block, with a gray top, marked "LDS-A3924RISI").
( ) LED2: "
( ) LED3: "
( ) LED4: "
( ) LED5: "
( ) LED6: "
( ) LED7: "
( ) LED8: "

That's it! All the parts should now be installed. Check your work carefully for unsoldered connections, bad solder joints, parts in backwards, or in the wrong places.

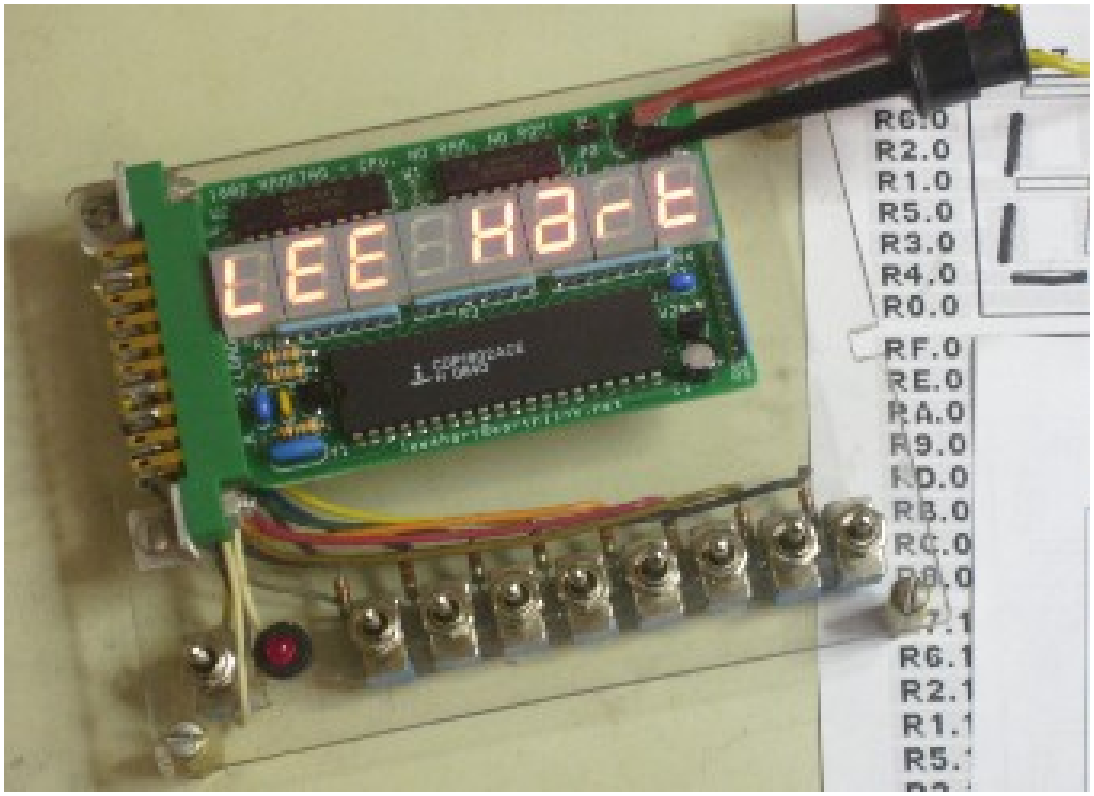
Testing

Connect a source of power between 4v and 6v to the pins of connector P1. Be sure to get the positive (+) and negative (-) correct! Now, plug the shorting jumper onto connector P2 (this is your "on-off switch". You should be rewarded with a random display of segments that changes about once per second, and repeats every 4 seconds. Success! :-)

At this point, the 1802 is simply displaying the random data in its registers from power-up. To load your own data, you need to build the "Programming Panel" shown on the schematic. The parts needed are listed in the Parts List on page 2 (see Note 1).

I used a piece of clear plastic, and mounted the switches, connector and LED to it. The parts are all hand-wired according to the schematic.

Note 1: I know, I know... the Front Panel should be included with the kit. But I built mine with parts I already had in my "junk box", and don't have any more. If I had to buy the switches and edge connector, the kit would be twice the price! If this kit is popular enough or I find some "great deals", maybe I can get the cost down. I'm also looking for ways to program it more simply with my Membership Card, a PC parallel port, or an Arduino etc. Got any ideas?



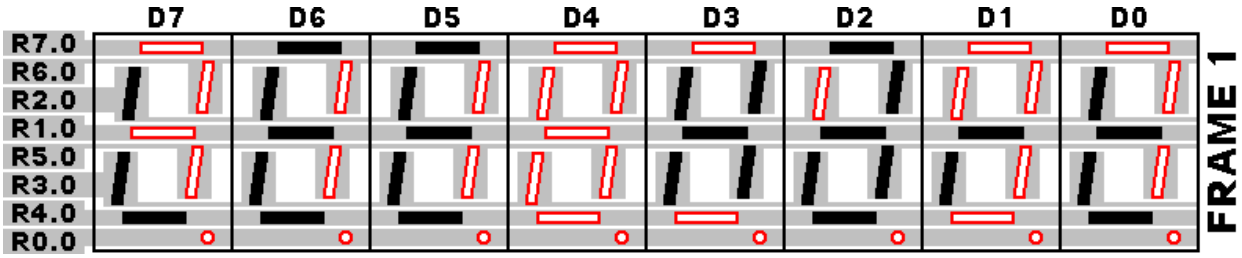
Programming

So how do we program this thing? To do that, we are going back... WAY back to the very beginnings of computer-time. Back to the days when there were no terminals, no assemblers, and no compilers. Just you, the computer, and its binary front panel with its switches and lights!

The Programming Panel is used to load your messages into the 1802's registers. Surprisingly, we don't even need the 1802's handy LOAD mode. Instead, the Nametag uses the CPU's WAIT line to hold it motionless until the next byte. This won't work with most microprocessors; but it does with the 1802! It will probably even work with the 1804/5/6 (versions of the 1802 that are otherwise difficult to use without an external ROM).

First, design the message you want to display. A worksheet is enclosed with blank spaces for each of the four 8-character messages. Use a pencil to fill in the segments that you want to be ON for each segment. (Pencil is better so you can correct mistakes.) You can print more copies of this worksheet from the web at http://www.sunrise-ev.com/MembershipCard/nametagmap.gif

For example, suppose you want the first message frame to be "LEE HART" like the photo above. The 7-segment display makes us get creative, and use lower-case letters for "a", "r" and "t". FRAME 1 on your worksheet would look like this:



See the long thin grey box on top? It says the top 8 horizontal segments are stored in R7.0 (the low byte of Register 7). Off is "0", and On (filled in) is "1". So R7.0 needs to be programmed with 0110 0100 binary, which is equal to 64 hexadecimal (or 64 hex for short).

First, we load 64 into the 1802. This is done with an LDI "Load Immediate" instruction (F8, 64). Then we move the 64 to the low half of register R7 with a PLO 7 "Put Low R7" instruction (A7). So, we will toggle in F8, 64, A7. This is done with Programming Panel switches S8-S0 like this:

	---S8---	S7------S0	---------- description ----------
1.	center	center	Start with all switches in the center-off position (RUN).
2.	FETCH	1111 1000 = F8	The 1802 fetches the LDI "Load Immediate" instruction.
3.	EXECUTE	0110 0100 = 64	Now set S7-S0 to the pattern that the LDI instruction loads.
4.	FETCH	1010 0111 = A7	The 1802 fetches the PLO 7 "Put Low R7" instruction.
5.	EXECUTE	center	S7-S0 are center-off because PLO doesn't need a 2nd byte.
6.	center	center	All switches back in the RUN position. The display scans, to show the results (top segments of frame 1 is correct).

Repeat this process for each of the eight registers for the first frame (see Note 2). Here is the rest of the data for the "LEE HART" message for the first frame. Note that at any time, you can set all switches to the center to RUN the program to see what you have entered so far:

	---S8---	S7------S0	---------- description ----------
	center	center	starting from the RUN position,
7.	FETCH	1111 1000 = F8	Load the pattern...
8.	EXECUTE	0000 1100 = 0C	...for the top right segments...
9.	FETCH	1010 0110 = A6	...into register R6.0
10.	EXECUTE	center	
11.	FETCH	1111 1000 = F8	Load the pattern...
12.	EXECUTE	1110 1001 = E9	...for the top left segments...
13.	FETCH	1010 0010 = A2	...into register R2.0
14.	EXECUTE	center	
15.	FETCH	1111 1000 = F8	Load the pattern...
16.	EXECUTE	0110 1111 = 6F	...for the center segments...
17.	FETCH	1010 0001 = A1	...into register R1.0
18.	EXECUTE	center	
19.	FETCH	1111 1000 = F8	Load the pattern...
20.	EXECUTE	0000 1100 = 0C	...for the bottom right segments...
21.	FETCH	1010 0101 = A5	...into register R5.0
22.	EXECUTE	center	
23.	FETCH	1111 1000 = F8	Load the pattern...
24.	EXECUTE	1110 1111 = EF	...for the bottom left segments...
25.	FETCH	1010 0011 = A3	...into register R3.0
26.	EXECUTE	center	
27.	FETCH	1111 1000 = F8	Load the pattern...
28.	EXECUTE	1110 0101 = E5	...for the bottom segments...
29.	FETCH	1000 0100 = 84	...into register R4.0
30.	EXECUTE	center	
31.	center	center	Flip all switches to RUN. You should see your message!

The second frame is loaded exactly the same way, except that the second digit N of all those An instructions (PLO or Put Low) uses n = 8 to F. The worksheet shows the register numbers for all the segments in each of the four frames.

The third frame is the same again, with the An instructions changed to Bn (PHI or Put High), and n = 0 to 7.

The fourth frame is also the same, with the An changed to Bn (PHI or Put High), and n = 8 to F.

The messages in each frame are completely separate. They can all be the same, or different. You can put the same message in successive frames, but with each one shifted 2 digits, so it scrolls left or right. You can repeat the message in two frames but with certain characters blank in one of them so they blink. Get creative!

After all that work to load them, you will be **DELIGHTED** to know that your messages will remain even when you remove jumper P2 to turn the Nametag off. As long as you leave a battery connected, the 1802 will remember the messages you put into it. Opening P2 puts it in "standby"; not off -- it cuts power to everything but the 1802. The 1802 draws essentially zero power in this state (less than 1 microamp), so the battery will last "forever".

> *Note 2: You don't need to enter anything for the last line of frames 1 and 3 (the decimal points). They use register R0, which happens to be the program counter. You can enable display of the decimal points by shorting jumper W2. Frames 2 and 4 will work. But in frames 1 and 3, the decimal points will just be a binary counter showing the current contents of the program counter: Interesting; but not very useful).*

Operation

Let's see if you were paying attention. Did you notice a pattern to those FETCH hex bytes? The first digit is the instruction (what the 1802 should do):

F n	Load n
A n	Put Low n
B n	Put High n

The second digit n tells the instruction what to do it to:

F 8	Load Immediate which means "use the byte immediately after this instruction".
A 1	Put Low in Register 1
B 3	Put High in Register 3

The entire 1802 instruction set works this way. This is one reason why it's such an easy microprocessor to use. It's also a key reason why this little project can work without any memory!

So how does it work? Where is the program? Where is the data? The schematic reveals the secret. The data is stored in the 1802's registers. There are sixteen 16-bit registers (32 bytes). That's enough to hold four message frames of 8 bytes each.

The program is produced by the unusual configuration of resistors between the data bus and power, ground, and the address bus. Each fetch cycle gets an opcode to read a register. Each

execute cycle displays that register's contents on its associated LED digit. As the program counter increments, each address is one higher, which produces an opcode one larger, which reads the next higher register number. The program (in hex) is thus 81, 82, 83... These are the instructions GLO R0, GLO R1, GLO R2... (Get Low register 0, Get Low register 1, Get Low register 2...). This reads and displays the lower byte of each of the 16 registers. At 8F (GLO RF) it advances to 90h, 91h, 92h... These happen to be the instructions GHI R0, GHI R1, GHI R2... (Get High register 0, Get High register 1, Get High register 2...). This displays the upper byte of each of the 16 registers. Tricky!

But wait; there's more. :-) Address bits A0, A1, and A2 scan the LED digits (with help from the 74HC138 decoder). Address bits A10 and A11 are wired so each 8-instruction sequence repeats 128 times before advancing to the next one. This makes each 8-character frame get displayed for a second before advancing to the next frame.

The 74HC241 octal buffer is actually being used as two 4-bit latches. I couldn't use a standard octal latch, because the high address bits need to be latched at TPA, and the low bits at TPB (and saved for the subsequent execute cycle).

This all works; but the LEDs would be dim because they are only active half the time (off during each fetch, and on during each execute cycle). So a simple wait-state generator is added, to make the execute cycles far longer than the fetches. The LEDs are thus active 99% of the time, for good brightness.

Future work

Toggle switches are about as simple as it gets; but they get old fast. They are also relatively expensive. Is there a better way to load the messages?

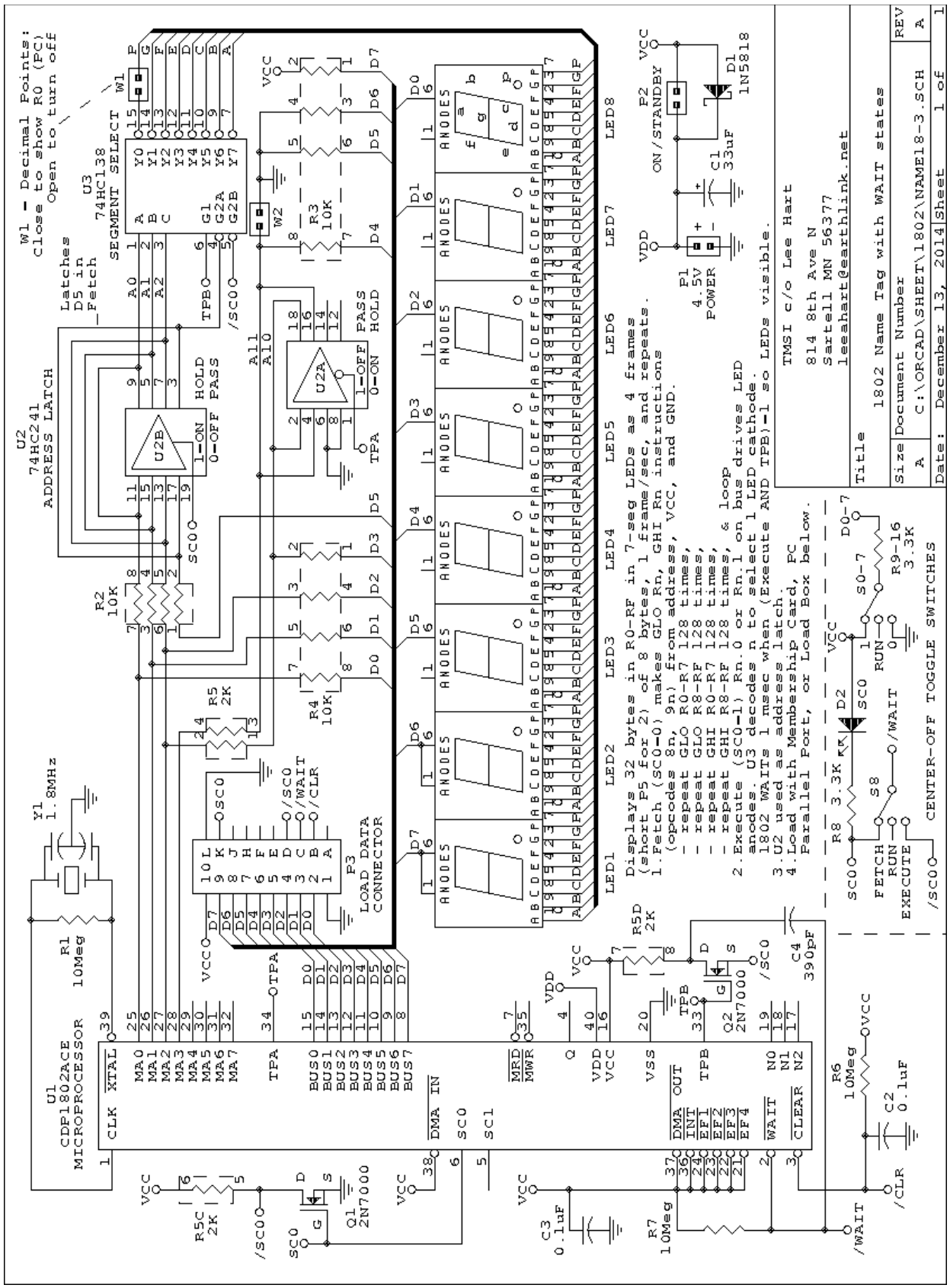
I know that my 1802 Membership Card can do it. But you have to make a little adapter board with a pair of 74HC244, -365, or -367 tri-state buffers to simulate the 9 switches. Then write a little program to load the desired messages into it auto-magically.

This same thing can be done with a traditional PC parallel port. I'm sure an Arduino, BASIC Stamp, or just about any other computer could be used as well.

But I wonder... what sort of creative ways can YOU think of to do it? Let me know what you come up with! :-)

I hope you enjoy this mad little project as much as I did. Have fun with it, and let me know what you think of it. So keep "running light without overbyte" as Dr. Dobbs used to say!

Lee A. Hart


W1 - Decimal Points:
Close to show R0 (PC)
Open to turn off
U2
74HC241
ADDRESS LATCH
Latches
D5 in
_Fetch
U3
74HC138
SEGMENT SELECT
W1
W2
U2B
1=ON
0=OFF
HOLD
PASS
U2A
1=OFF
0=ON
PASS
HOLD
TPA
TPB
/SC00
SC00
R2
10K
R3
10K
R4
10K
R5
2K
R5C
2K
R5D
2K
R6
10Meg
R7
10Meg
R8
3.3K
R9-16
3.3K
R1
10Meg
Y1
1.8MHz
C1
33uF
C2
0.1uF
C3
0.1uF
C4
390pF
D1
1N5818
P1
4.5V
POWER
P2
ON/STANDBY
VDD
VCC
P3
LOAD DATA
CONNECTOR
/SC0
/WAIT
/CLR
Q1
2N7000
Q2
2N7000
U1
CDP1802ACE
MICROPROCESSOR
CLK
XTAL
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
BUS0
BUS1
BUS2
BUS3
BUS4
BUS5
BUS6
BUS7
MRD
MWR
VDD
VCC
VSS
TPB
DMA OUT
INT
EF1
EF2
EF3
EF4
WAIT
CLEAR
N0
N1
N2
DMA IN
SC0
SC1
Q
LED1
LED2
LED3
LED4
LED5
LED6
LED7
LED8
ANODES
Displays 32 bytes in R0-RF in 7-seg LEDs as 4 frames
(short P5 for 2) of 8 bytes, 1 frame/sec, and repeats.
1.Fetch (SC0-0) makes GLO Rn, GHI Rn instructions
(opcodes 8n, 9n) from address, VCC, and GND.
- repeat GLO R0-R7 128 times,
- repeat GLO R8-RF 128 times,
- repeat GHI R0-R7 128 times,
- repeat GHI R8-RF 128 times, & loop
2.Execute (SC0-1) Rn.0 or Rn.1 on bus drives LED
anodes. U3 decodes n to select 1 LED cathode.
1802 WAITs 1 msec when (Execute AND TPB)-1 so LEDs visible.
3.U2 used as address latch.
4.Load with Membership Card, PC
Parallel Port, or Load Box below.
D0-7
S0-7
S8
FETCH
RUN
EXECUTE
CENTER-OFF TOGGLE SWITCHES
TMSI c/o Lee Hart
814 8th Ave N
Sartell MN 56377
leeahart@earthlink.net
Title
1802 Name Tag with WAIT states
Size
A
Document Number
C:\ORCAD\SHEET\1802\NAME18-3.SCH
REV
A
Date: December 13, 2014
Sheet 1 of 1

FRAME 1
FRAME 2
FRAME 3
FRAME 4
D0
D1
D2
D3
D4
D5
D6
D7
R7.0
R6.0
R2.0
R1.0
R5.0
R3.0
R4.0
R0.0
RF.0
RE.0
RA.0
R9.0
RD.0
RB.0
RC.0
R8.0
R7.1
R6.1
R2.1
R1.1
R5.1
R3.1
R4.1
R0.1
RF.1
RE.1
RA.1
R9.1
RD.1
RB.1
RC.1
R8.1