Tim Ahrens WA5VQK
7405 Ladybug St.
Austin TX 78744

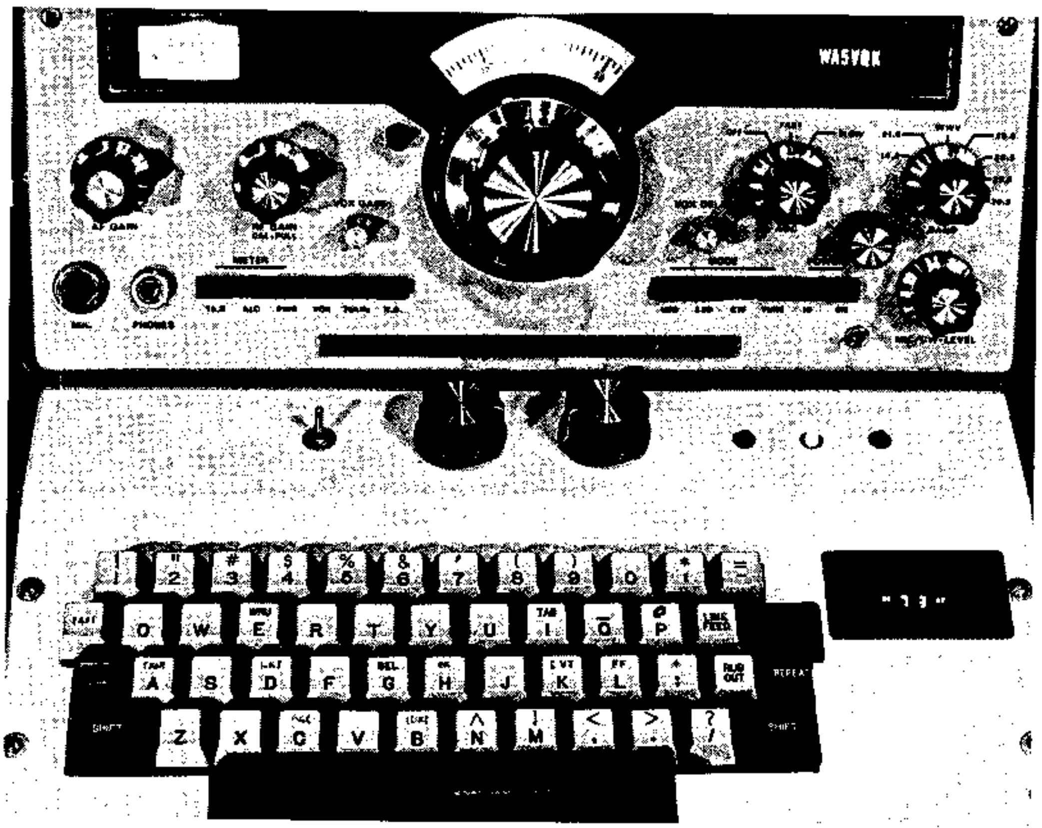
The Perfect Morse Machine

—send and receive CW with a dedicated micro

Author's note: I will supply a photocopy of the PC board artwork for $4.00. I will also program reader-supplied 2708s for $6.00. Both are payable by money order or certified check. I also wish to thank Mike Hadley WA7NLM for his audio filter design.

Have you seen all of the ads coming out for "automatic" Morse keyboards and readers with one big alphanumeric "eye" that gives you a readout as someone is sending code? Then there are the "complete" stations which will do both on a TV screen for only $600. The item I am about to describe can do all of this and more, with many possible options, for less than $100.

About three years ago at the Dayton Hamvention, I picked up a program for a computer-controlled Morse code station. This program was written with the MC6800 computer in mind and was fathered by Don Jackson W7GKU and Jim Bainter WA7VKZ. The computer and the program in our story have changed significantly, but still are based on their algorithms. The original program was designed for use with the old Mikbug® evaluation module, and used the onboard ACIA (Asynchronous Communications Interface Adapter) serial port for communications with a TTY or CRT terminal.



Well, after I finally got to the point where I had a little knowledge of the 6800, I decided to try to make this thing work. Keep in mind that a little knowledge is dangerous! I had to do some work, but after a while, a wire-wrapped version appeared on a card within my computer. I didn't believe it, but it really worked. The only problem I had with it was the way it copied code. It was so perfect that all of those with sorry fists would really mess it up. With all of the means available to generate decent code, whether keyer or keyboard, there is really no excuse for a poor fist.

Well, enough of my soapbox. With this gizmo, everybody can have perfect CW capabilities. After the original version, wire-

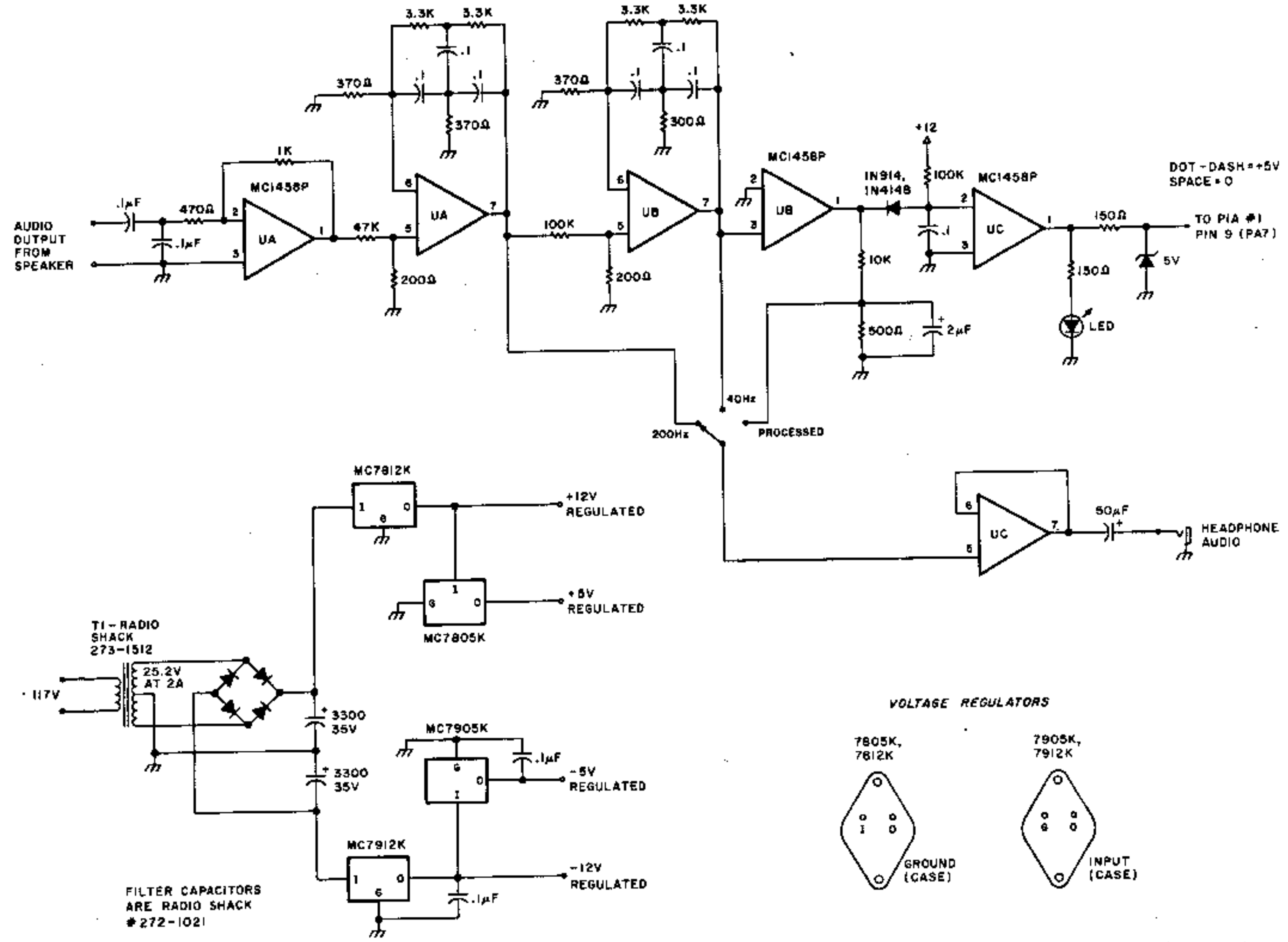


Fig. 1. This is the audio filter and recommended power supply. The input audio is filtered by the op amps and turned into dc to be fed into the PIA. Vcc = pin 8 (+12); Vee = pin 4 (−12).

wrapped deep within the bowels of my mainframe, worked, the next thought was to make it a fully self-contained system with a dedicated CPU to work with. The most logical choice was to use the MC6802—son of 6800. With clock and RAM all on silicon, at least three packages could be saved over the original 6800 design. The age of the small dedicated system has arrived.

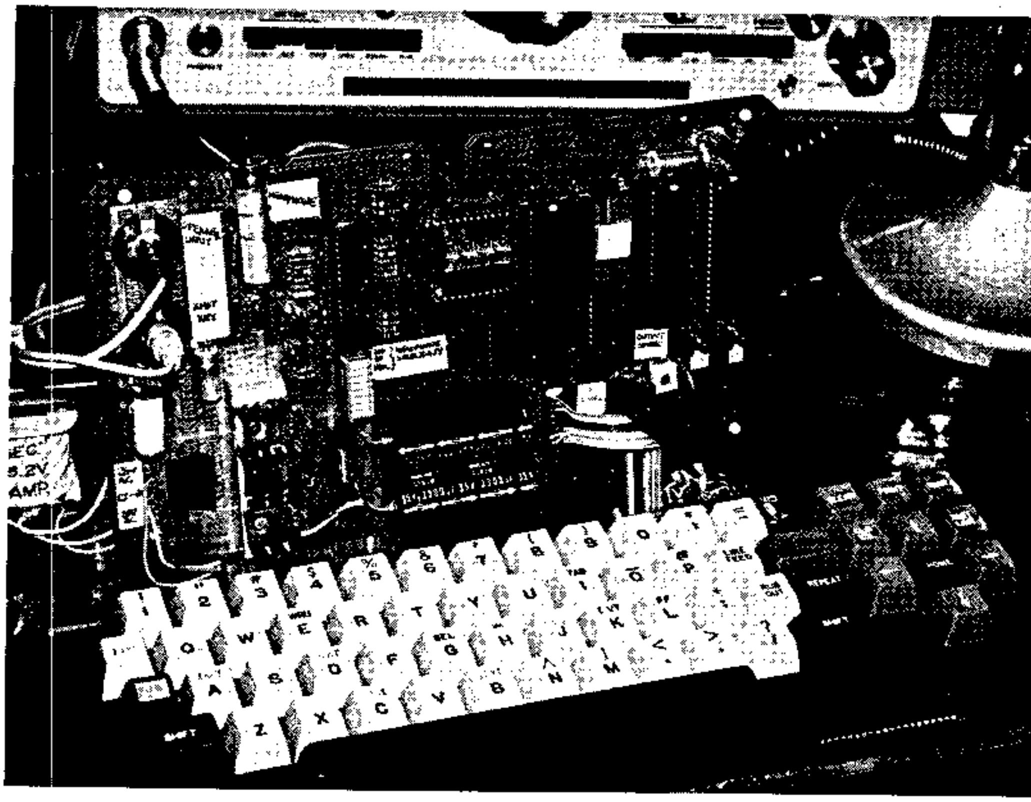
The following evening was spent on wire-wrapping a small board with the system. The program was, of course, stored in an EPROM for automatic "boot" and for more additions later. The two biggest drawbacks that I could find were that it still required a terminal which was both large and expensive, and it had a switch to go from receive to transmit, and so forth. The last time I saw one of those things was as a Novice with my Globe Scout and S100!

So, these little things had to be fixed. OK, the switch could be replaced with a different polling routine within the software. No sweat, but what about the terminal? Well, plugging in a keyboard would be no trouble, but something to peer at was also required. What to do? Well, Burroughs makes some fine display panels which can display about 30 characters on a line, but you might have to float a loan to buy one—and having to use 250 volts didn't appeal to me, anyway.

Enter the DL-1416. Litronix has done it again. The 1416 is a four-digit, 16-segment, alphanumeric intelligent display, which, when fed parallel ASCII, will display the most-used 64 characters of the ASCII set. It is a five-volt-only device, and by placing it directly on the CPU bus, will appear as "write-only RAM." Unfortunately, the slow setup times and the need to have data valid before Read/Write comes out present a few problems. The first version had the LEDs just hanging on the data bus and worked with some degree of reliability. But when Rs started coming out as Ps, I knew that the timing specifications were being violated.

After much manipulation of the clock and stretching of the E pulse, I determined that too much glue (peripheral hardware) would be required to just leave it on the bus. Then I remembered that only half of PIA 2 was being used, so the most logical choice was to put the 1416 on the peripheral data bus. By setting the control register within the PIA first to a \$34 and then to a \$3C, the CB2 line would go low and come high again, all done while the data was latched onto the output lines. Great—use this for the $\overline{W}$ (write) line, and tie it to $\overline{CE}$ (chip enable), too! The timing of the 1416 specifies that the falling edge of $\overline{CE}$ must be at least 500 ns before the rising edge of $\overline{W}$. So having them both come in at the same time is permissible. See the timing diagrams.

According to Appnote 9A from Litronix, systems which use only a 6-bit



ASCII code can still utilize the 1416 by inverting D5 and feeding it into D6. By doing it this way, two "spare" data lines on the PIA are saved. These two lines are used for addressing either of the four digits to be written to. All of the work writing to the LEDs is now being accomplished in software.

The clock within the 6802 is really a strange beast. It can accept any parallel-resonant crystal from 1 to 4 MHz. Of course, the crystal is divided by four, so remember this when selecting your crystal. In this program, the CPU will work fine even when running at its minimum frequency (100-kHz bus speed, 400-kHz crystal frequency).

A funny thing happened

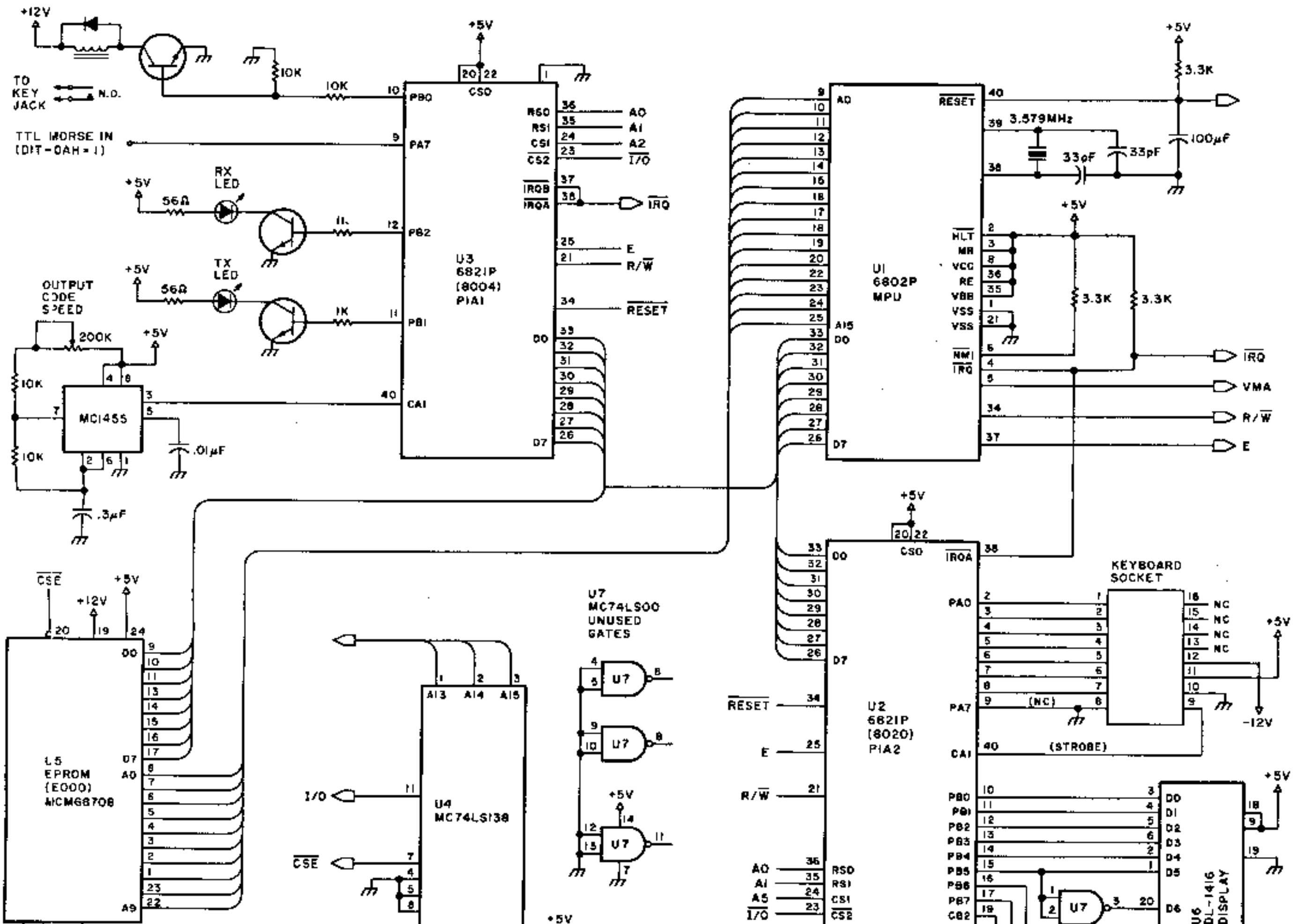


Fig. 2. The main processor, composed of the MC6802 MPU, two PIAs, and an EPROM.

while experimenting with different crystal types; the thing oscillated even without a crystal! The frequency was really slow and quite unstable, but it worked. Even after power down and up, it continued to oscillate. Since the 6802 is not specified for use without a crystal, Motorola won't guarantee it to work without one, but, just for fun, why not try it out on yours when you build it?

You might think that having only four digits a little bigger than a calculator display would be a real hassle to use. Not so! The program takes the incoming data or keyboard data and puts it in the right-hand display. As the next character comes in, it is shifted left one digit. Instant "Times Square" display. I was a bit hesitant to use on-

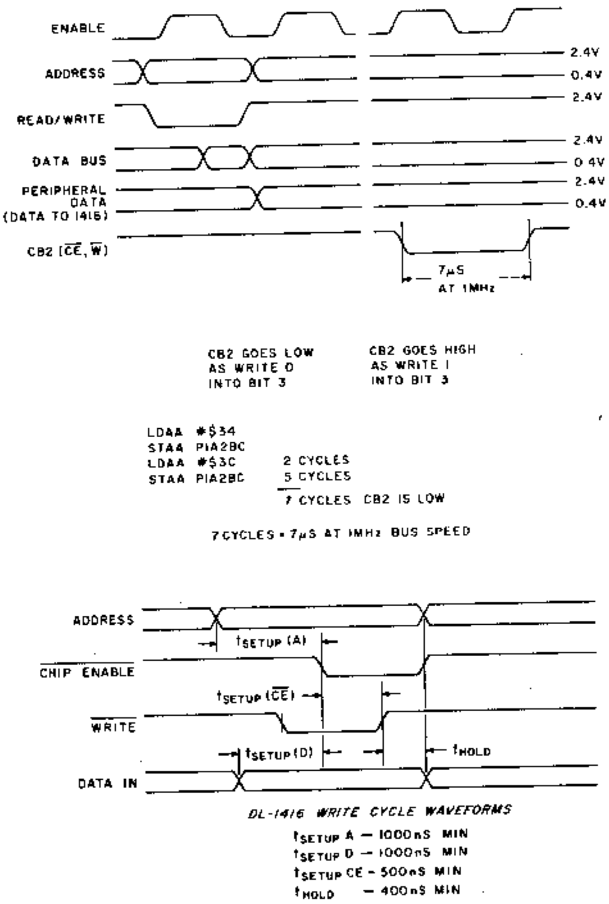


Fig. 3. PIA write-timing to displays, and Litronix-recommended write-cycles for the DL-1416.

POWER SUPPLY NOTES

The total current requirements for the reader/talker are given below. These figures are the absolute maximums for the individual components, so don't expect them to be that high, but, just in case, go ahead and plan for it.

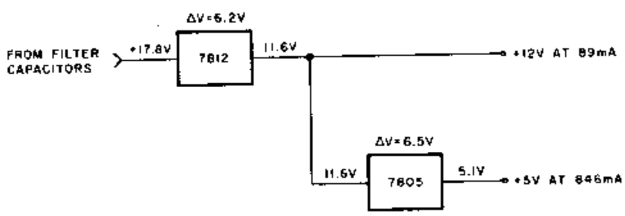
Current requirements for the reader/talker. All values are in milliamps and are absolute maximums.

	+5 V	+12 V	−12 V	−5 V
MC6802	240	—	—	—
MC6821	110	—	—	—
MC6821	110	—	—	—
MCM68708	10	65	—	65
MC74LS138	6.4	—	—	—
MC74LS00	15	—	—	—
DL1416	100	—	—	—
MC1458	—	8	8	—
MC1458	—	8	8	—
MC1458	—	8	8	—
3 LEDs	30	—	—	—
Average Keyboard	225	—	45	—
Maximum Totals	846.4	89	69	45

As can be seen in the totals, all currents can be handled with the three-terminal-style regulators. The following calculations are to determine what the power dissipation for each device is. For the three-terminal regulators, the power dissipation can be expressed by the following equation:

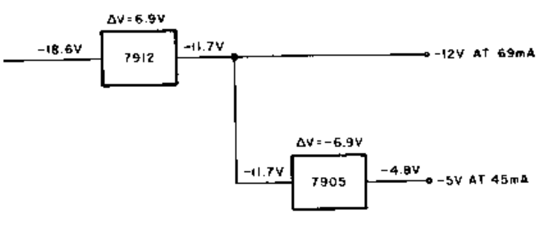
$P_D = (\Delta E_{IN}) I_{OUT} + E_{IN} (I_Q)$, where: P_D = Power dissipation, E_{IN} = In-Out voltage, I_{OUT} = output current, and I_Q = current through ground lug.

For the positive voltage regulators, the following schematic is used, and the voltages are labeled. These were typical, using the values of transformers and capacitors specified in Fig. 1.



$$7812\ P_D = (6.2\ V)(.935\ A) + 17.8\ (.008) = 5.94\ W$$
(Note: .935 A = .846 A + .089 A)
$$7805 P_D = (6.5\ V)(.846\ A) + 11.6\ (.008) = 5.34\ W$$

In the same light, the minus voltage regulators can be also calculated:



$$7912\ P_D = (6.9\ V)(.114\ A) + (18.6)(.008) = .935\ W$$
$$7905\ P_D = (6.9\ V)(.045\ A) + 11.7\ (.008) = .404\ W$$

By taking the input of the 7805 from the output of the 7812, the power dissipation of the 7812 was raised from about ½ Watt to almost 6 Watts. But by taking the 7805 input directly at the capacitor output, its P_D would have been 10.75 Watts! This would be very difficult to heat sink, so by doing it as shown, the total power dissipation is about the same, only divided between two parts.

According to the *Motorola Linear IC Data Book*, to dissipate the required power (5.9 and 5.3 W), a heat sink which can dissipate 15°C/Watt will be required. The minus voltage regulators are not as critical, as they have a lesser current demand on them. I recommend the use of the -K suffix regulators (TO-3 case), because their thermal resistance (junction-to-case vs. junction-to-air in the plastic package) is so much better. Heat sink them with an appropriate sink, preferably to a metal case, and no problems should be encountered. The negative regulators' requirements can be satisfied by simply bolting them to the chassis. (Be sure to insulate their cases!)

The voltage regulators shown in the pictures are of the plastic style and get quite hot even with the heat sinking shown. When the project is put into an enclosure, TO-3-style regulators will replace them.

ly four digits, but after using it, I find that 4 is really not too few, and the faster the code is sent, the easier it is to read. The new design now automatically transfers from receive to transmit and back again after all characters have been sent. The algorithm used in the code conversion is set up so that it automatically senses speed changes and adjusts itself for the correct speed. This can be anywhere between about one-half wpm to 300 wpm. Changing speed requires only about 2 characters to be lost, so it is quite efficient. Neat, eh?

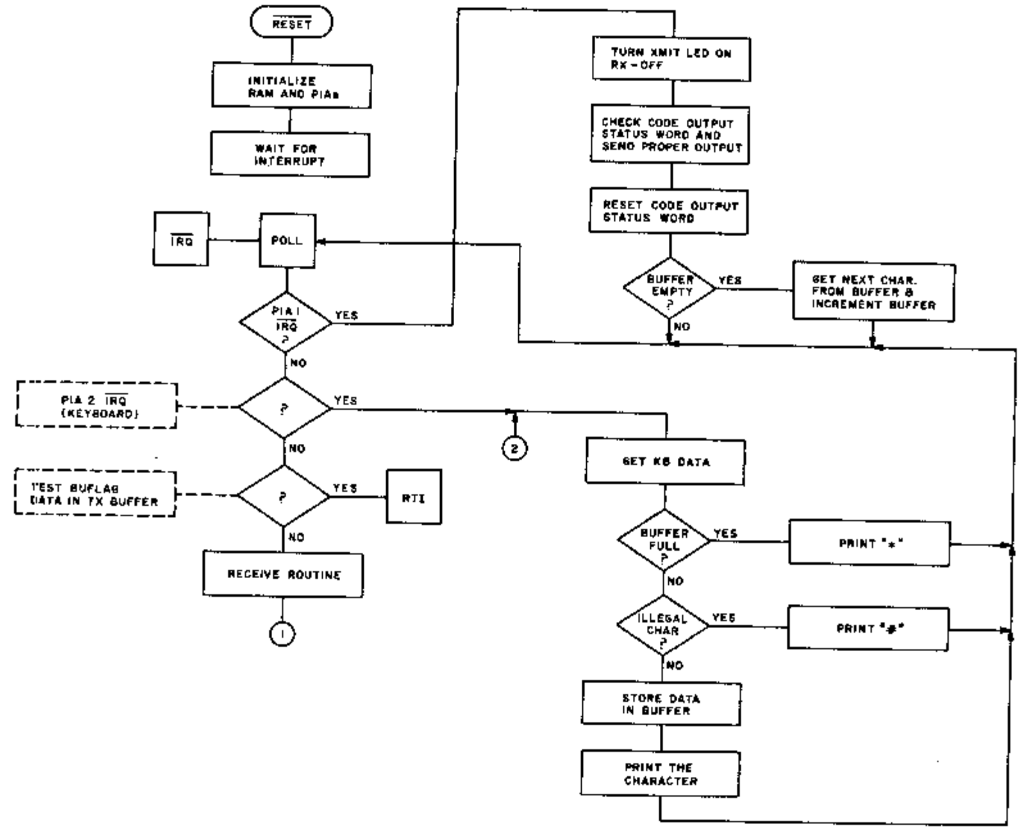


Fig. 4. Flowchart, part 1.

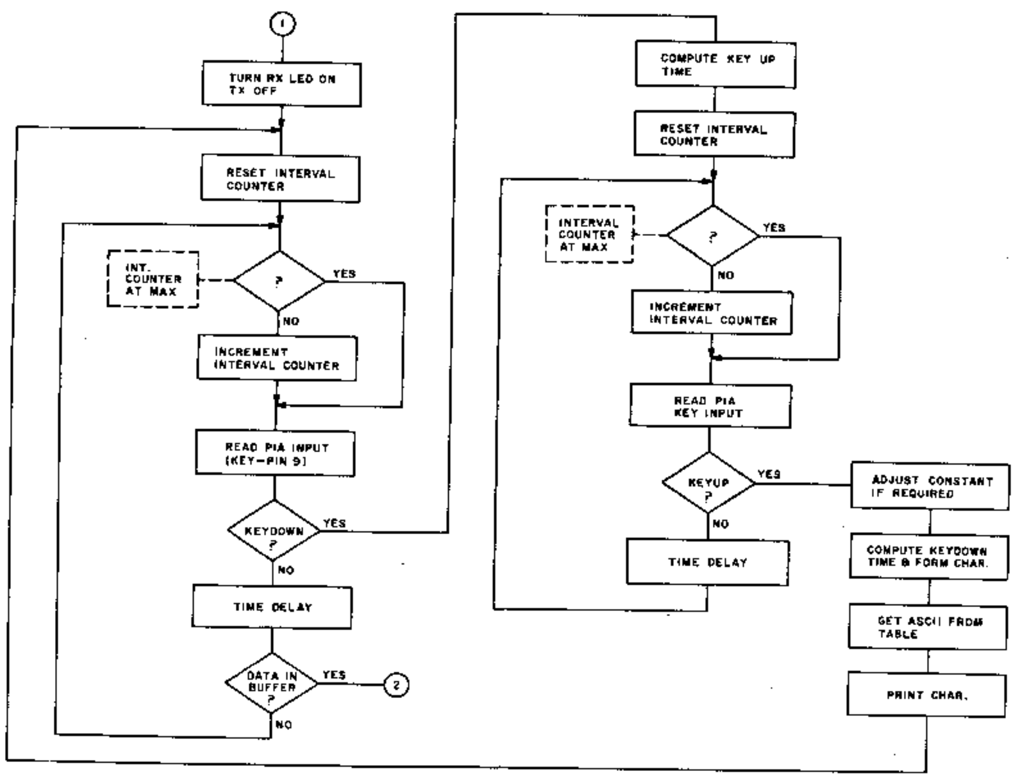


Fig. 5. Flowchart, part 2.

Back before I built the keyboard/reader, I thought that it would be "cheating" to use one; it had been done manually for such a long time, who was I to change any preconceived ideas? Anyway, I have found just the opposite to be true! Number 1, the computer is not foolproof. It can be messed up by zero-beat QRM, sloppy fists, or high noise levels. So, when you start your QSO, try not to get carried away and go too fast, because if one of these should happen, you might be stranded by sending faster than you can receive. Another thing is that you are seeing a character appear on the display at the same time you are hearing it. No better way for associative learning than to get the code into your eyes and ears simultaneously. I've found my speed to be dramatically increasing following the use of it for a couple of weeks. Also, you have a tendency to follow the conversation along in your head—the only way to do it for above 15 or 20 wpm. The final plus is that you hear perfect code being sent by your keyboard, and it makes you wish that everyone had one of these things! So, it's not really cheating, but only another progressive learning device provided by the computer.

As this design recognizes a single input to determine a character and a single output for code, why couldn't it also be used for Baudot or ASCII teletype™? This is just a bug in your ear—look for some further programs in later articles. Wouldn't it be nice to have a truly portable TTY "machine"?

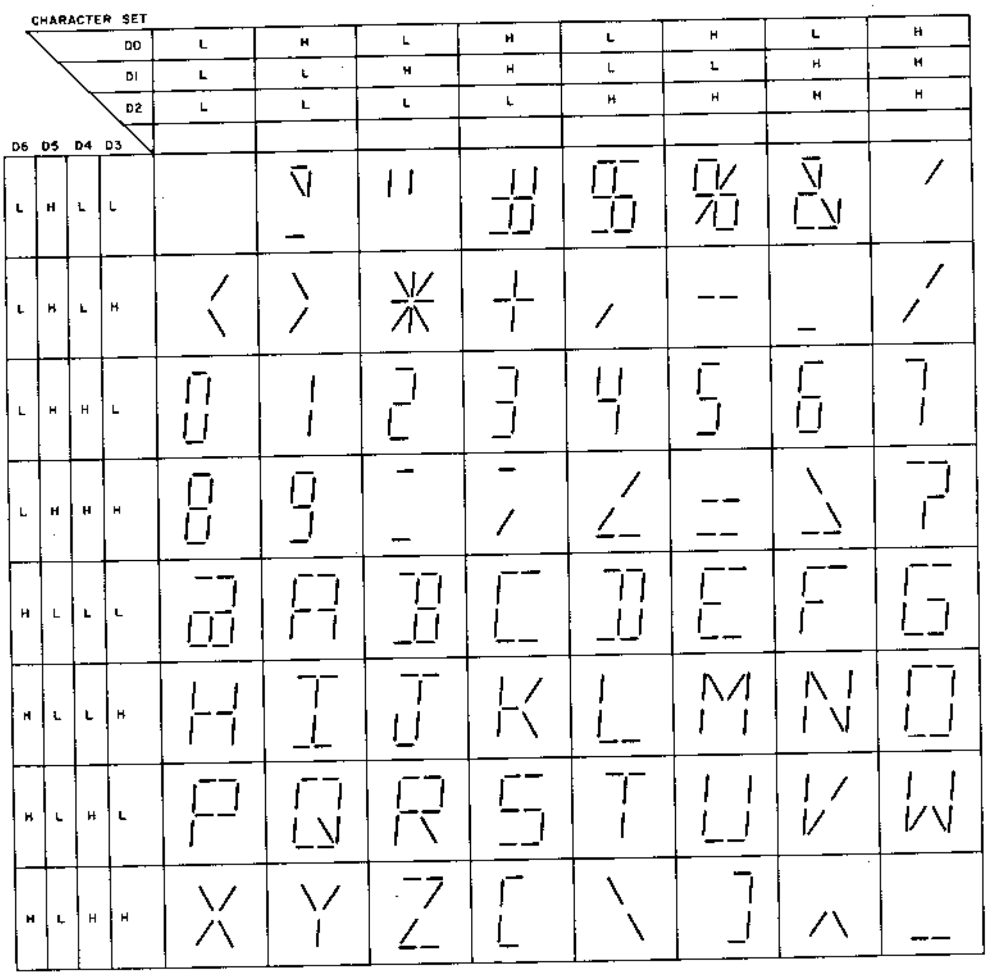
Construction

As you can see in Fig. 1,

the audio from a headphone jack or speaker tie-in goes into a filter-processor where it is converted from audio tones to TTL-compatible levels. This audio input is fed into an op amp preamplifier, so not a whole lot of audio is required. You might want to pad it if your receiver overdrives the device. You also can plug your headphones into the jack on the board and use it for a filter. The first switch position provides selectivity down to about 200 Hz, the second to 40 Hz, and the third to processed audio—this is a really fun feature. After it is tuned in correctly, all you hear is a tone. No QRM, no static, no garbage, only a pure representation of the sent character. This is a really weird feeling, especially if you have never used it before. If you don't want to build the computer, at least build the filter—it works great for CW by itself.

After this processed audio is converted to a voltage level, it is fed into a single bit of a PIA (Peripheral Interface Adapter) parallel port. Two of these PIAs are used on the board, one for Morse in and out, transmit speed timing, and indicator LEDs. The second PIA is for the keyboard and LED display unit. The 555 circuit is set for the desired speed at which you wish to transmit. In this design, a relay is provided for Morse code output. Although my HW-104 uses positive voltage keying and worked great with just a single keying transistor, a reed relay provides a more versatile interface for use with any type of keying, from grid-block to cathode.

The EPROM in the circuit is used to hold the pro-



LOADING DATA

$\overline{CE}$	$\overline{CU}$	$\overline{W}$	ADDRESS A_1	A_0	DATA INPUT D6	D5	D4	D3	D2	D1	D0	DIGIT 3	DIGIT 2	DIGIT 1	DIGIT 0
H	X	X	X	X	X	X	X	X	X	X	X	N/C	N/C	N/C	N/C
L	H	L	L	L	H	L	L	L	L	L	H	N/C	N/C	N/C	A
L	H	L	L	H	H	L	L	L	L	H	L	N/C	N/C	B	A
L	H	L	H	L	H	L	L	L	L	H	H	N/C	C	B	A
L	H	L	H	H	H	L	L	L	H	L	L	D	C	B	A
L	H	L	L	L	H	L	L	L	H	L	H	D	C	B	E
L	H	L	H	L	H	L	L	H	L	H	H	D	K	R	E
L	H	L	—	—	—	—	—	—	—	—	—	SEE CHARACTER SET			

LOADING CURSOR

$\overline{CE}$	$\overline{CU}$	$\overline{W}$	ADDRESS A_1	A_0	DATA INPUT D6	D5	D4	D3	D2	D1	D0	DIGIT 3	DIGIT 2	DIGIT 1	DIGIT 0
H	X	X	X	X	X	X	X	X	X	X	X	D	K	B	E
L	L	L	X	X	X	X	X	L	L	L	H	D	K	B	.
L	L	L	X	X	X	X	X	L	L	L	L	D	K	B	E
L	L	L	X	X	X	X	X	L	L	H	L	D	K	.	E
L	L	L	X	X	X	X	X	L	H	L	L	D	.	B	E
L	L	L	X	X	X	X	X	H	L	L	L	.	K	B	E
L	L	L	X	X	X	X	X	H	H	H	H	.	.	.	.
L	L	L	X	X	X	X	X	L	L	L	L	D	K	B	E

Fig. 6. Litronix DL-1416 character set and truth table. X = don't care; N/C = no change.

Program listing. Software for reader in 6800 assembler source code.

```
                                   NAM    MORSEMAX (LED VERSION)
A 0000                             ORG    $0000
                     *****************************************************
                     * MORSE CODE/ASCII SEND/RECEIVE PROGRAM
                     * FOR THE MOTOROLA 6802 MICROPROCESSOR
                     * FROM AUSTIN, TEXAS
                     * MICROPROCESSOR CAPITOL OF THE WORLD!
                     *****************************************************
                     * I/O HARDWARE CONFIGURATION:
                     *     PIA1 ADDRESS - $8004
                     *        CA1 - INTERRUPT TIMING FOR XMIT CODE,
                     *              2-50 HZ.
                     *        PA7 - RECEIVE CODE INPUT
                     *        PB0 - CODE OUT
                     *        PB1 - TRANSMIT LED
                     *        PB2 - RECEIVE LED
                     *     PIA2 ADDRESS (KBD) - $8020
                     *     LITRONIX DISPLAY DL-1416 4 DIGIT LED - $C000
                     *     MCM2708L EPROM - $E000
                     *****************************************************
                     * ALL COMMONLY USED MORSE CHARACTERS
                     * ARE AVAILABLE:
                     *
                     *       SPACE - SPACE
                     *       ESC   - AS
                     *       =     - BT
                     *     CNTRL A - KN
                     *           B - BK
                     *           C - AR
                     *           D - SK
                     *           F - SN
                     *           H - ERROR ( 8 DOTS )
                     *
                     *****************************************************
                     *
                     *** TEMPORARY STORAGE FOR VARIABLES AND BUFFER **
                     *
A 0000    0002  A CVCX    RMB    2          INDEX REG CONVERT STORE
A 0002    0002  A SAVEX   RMB    2          X-REG TEMP STORAGE
A 0004    0001  A COUNT   RMB    1
A 0005    0001  A RESMSK  RMB    1          COSTA RESET MASK
A 0006    0001  A BUFLAG  RMB    1          B7=1  DATA IN BUFFER
A 0007    0003  A TMPSVE  RMB    3          TEMP SAVE AREA FOR LED DIS
A 000A    0001  A COSTA   RMB    1          CODE OUTPUT STATUS
                  *B7 DIT FLAG,B6 DAH FLAG,B5 ELEMENT SPACE FLAG
                  *B4 WORD SPACE FLAG,B3 CHAR. SPACE FLAG
A 000B    0001  A LETYPE  RMB    1          LAST ELEM TYPE DOT=0 DASH
A 000C    0001  A HLETIM  RMB    1          HALF LAST ELEM TIME
A 000D    0001  A TLETIM  RMB    1          TWICE LAST ELEM TIME
A 000E    0001  A SPEEDK  RMB    1          SPEED CONSTANT
A 000F    0001  A RCHAR   RMB    1          CHAR BEING RECEIVED
A 0010    0001  A LDATIM  RMB    1          LAST DASH TIME
A 0011    0001  A TQLDAT  RMB    1          3/4 LAST DASH TIME
A 0012    0001  A TLDAT   RMB    1          TWICE LAST DASH TIME
A 0013    0001  A KDTIM   RMB    1          KEYDOWN INTERVAL TIME
A 0014    0001  A KUTIM   RMB    1          KEYUP INTERVAL TIME
A 0015    0001  A CHCTR   RMB    1          REC. CHARACTER COUNTER
A 0016    0002  A RECX    RMB    2          REC. INDEX REG. TO ACCA.
A 0018    0055  A BUFBOT  RMB    85         XMIT BUFFER BOTTOM
A 006D    0001  A BUFTOP  RMB    1          XMIT BUFFER TOP

                  * PIA USED FOR I/O OF CW AND LED STATUS
A 8004                    ORG    $8004
A 8004    0001  A PIA1AD  RMB    1
A 8005    0001  A PIA1AC  RMB    1
A 8006    0001  A PIA1BD  RMB    1
A 8007    0001  A PIA1BC  RMB    1

                  * PIA USED FOR KEYBOARD INPUT AND LED DISPLAY OUT
A 8020                    ORG    $8020
A 8020    0001  A PIA2AD  RMB    1          *KEYBOARD
A 8021    0001  A PIA2AC  RMB    1
A 8022    0001  A PIA2BD  RMB    1          *DISPLAY
A 8023    0001  A PIA2BC  RMB    1

          007F  A STACK   EQU    $007F
          FFF8  A IRQVEC  EQU    $FFF8
                  *
A E000                    ORG    $E000
                  *                KN  BK  AR  SK   ACK
A E000    00    A CODE    FCB    0,$B4,$8B,$54,$16,0,$14,0
                  *             ERR           SP
A E008    01    A         FCB    1,0,0,0,0,$21,0,0
A E010    00    A         FCB    0,0,0,0,0,0,0,0
                  *                     AS
A E018    00    A         FCB    0,0,0,$44,0,0,0,0
                  *               SP     "
A E020    21    A RTAB    FCB    $21,0,$4A,0,0,0,0,$7A
                  *               (   )        ,  -  .  /
A E028    B6    A         FCB    $B6,$B2,0,0,$CE,$86,$56,$94
                  *               0   1   2   3   4   5   6   7
A E030    FC    A         FCB    $FC,$7C,$3C,$1C,$0C,4,$84,$C4
                  *               8   9   :   ;       BT(=)  ?
A E038    E4    A         FCB    $E4,$F4,$E2,$AA,0,$8C,0,$32
                  *                 A   B   C   D   E   F   G
A E040    00    A         FCB    0,$60,$88,$A8,$90,$40,$28,$D0
                  *               H   I   J   K   L   M   N   O
A E048    08    A         FCB    $08,$20,$78,$B0,$48,$E0,$A0,$F0
                  *               P   Q   R   S   T   U   V   W
A E050    68    A         FCB    $68,$D8,$50,$10,$C0,$30,$18,$70
                  *               X   Y   Z
A E058    98    A         FCB    $98,$B8,$C8

                  ***RESTART ROUTINE***
A E05B CE 007F  A RESRT   LDX    #$7F       CLR RAM 0-7F
A E05E 6F 00    A L1      CLR    0,X
A E060 09                 DEX
A E061 26 FB E05E         BNE    L1
A E063 CE 2020  A         LDX    #$2020
A E066 DF 07    A         STX    TMPSVE     'CLEARS TMPSVE'
A E068 DF 08    A         STX    TMPSVE+1
A E06A CE 0F01  A         LDX    #$0F01
A E06D DF 0E    A         STX    SPEEDK     INZ SPEED CONS & RCHAR
A E06F 86 80    A         LDAA   #$80
A E071 97 6D    A         STAA   BUFTOP     INZ BUFTOP
A E073 09                 DEX
A E074 CE E000  A         LDX    #CODE
A E077 DF 00    A         STX    CVCX       INZ CVCX
A E079 CE 006D  A         LDX    #BUFTOP
A E07C DF 02    A         STX    SAVEX
A E07E CE E0B4  A         LDX    #POLL      INZ IRQ VECTOR
A E081 FF FFF8  A         STX    IRQVEC
A E084 CE 8004  A         LDX    #PIA1AD
A E087 6F 01    A         CLR    1,X        CLR PIA1AC
A E089 6F 03    A         CLR    3,X        CLR PIA1BC
A E08B 8E 0007  A         LDS    #$0007     CA1=+ & ALLOWED
A E08E AF 00    A         STS    0,X        (PIA1AD & PIA1AC)
A E090 8E FF34  A         LDS    #$FF34
A E093 AF 02    A         STS    2,X        (PIA1BD & PIA1BC)
A E095 A6 00    A         LDAA   0,X        CLR IRQA FLAGS
A E097 A6 02    A         LDAA   2,X        CLR IRQB FLAGS
A E099 86 00    A         LDAA   #$00
A E09B B7 8020  A         STAA   PIA2AD
A E09E 86 07    A         LDAA   #$07       CA1 NEG INPUT
A E0A0 B7 8021  A         STAA   PIA2AC
A E0A3 86 FF    A         LDAA   #$FF
A E0A5 B7 8022  A         STAA   PIA2BD     SETS PIA2B FOR OUTPUTS
A E0A8 86 04    A         LDAA   #$04
A E0AA B7 8023  A         STAA   PIA2BC     SETS DDR
A E0AD 8E 007F  A         LDS    #STACK     INZ STACK POINTER
A E0B0 0E                 CLI               CLEAR INTERRUPT FLAG
A E0B1 3E         EXEC    WAI
A E0B2 20 FD E0B1         BRA    EXEC
                  *
                  ***JUMP FROM IRQ VECTOR***
A E0B4 7D 8005  A POLL    TST    PIA1AC
A E0B7 2B 0E E0C7         BMI    POLL2      (MC1455 IRQ)
A E0B9 7D 8021  A         TST    PIA2AC
A E0BC 2B 6A E128         BMI    COMMR1     (KEYBOARD IRQ)
A E0BE 7D 0006  A         TST    BUFLAG
A E0C1 2B 03 E0C6         BMI    NOTRCV     (DATA IN BUFFER)
A E0C3 7E E1E9  A         JMP    REC
A E0C6 3B         NOTRCV  RTI

                  ***TRANSMIT ROUTINE***
A E0C7 F6 8004  A POLL2   LDAB   PIA1AD
A E0CA F6 8006  A         LDAB   PIA1BD
A E0CD CA 02    A         ORAB   #2         XMIT LED ON
A E0CF C4 FB    A         ANDB   #$FB       REC LED OFF
A E0D1 F7 8006  A         STAB   PIA1BD
A E0D4 96 0A    A         LDAA   COSTA      GET CODE OUTPUT STATUS
A E0D6 2A 07 E0DF         BPL    CFDAH      CHECK FOR DAH
A E0D8 C6 7F    A         LDAB   #$7F
A E0DA D7 05    A         STAB   RESMSK     DIT RESET
A E0DC 5F         CES1    CLRB              DIT BEING SENT
A E0DD 20 29 E108         BRA    CKCNT      TEST COUNT
A E0DF 48         CFDAH   ASLA              TEST FOR DAH
A E0E0 2A 08 E0EA         BPL    CFES       CHECK FOR ELEMENT SPACE
A E0E2 C6 BF    A         LDAB   #$BF
A E0E4 D7 05    A         STAB   RESMSK     DAH RESET
A E0E6 C6 04    A         LDAB   #4         DAH BEING SENT
A E0E8 20 1E E108         BRA    CKCNT

A E0EA 48         CFES    ASLA              TEST FOR ELEMENT SPACE
A E0EB 2A 0A E0F7         BPL    CFWS       CHECK FOR WORD SPACE
A E0ED C6 DF    A         LDAB   #$DF
A E0EF D7 05    A         STAB   RESMSK     ELEMENT SPACE RESET
A E0F1 8D 2C E11F         BSR    SOZERO
A E0F3 C6 01    A         LDAB   #1
A E0F5 20 11 E108         BRA    CKCNT
```

gram permanently until it is desired to erase it. For a one-time shot, specify the plastic version. Much less expensive, but good only once. On the ceramic types, be sure to keep the adhesive paper over the quartz lid, as it might start to forget if exposed to ultraviolet light. The MC6802 processor contains an on-board clock generation circuit and 128 bytes of RAM. This RAM is used by the processor for temporary storage and for the 85-character keyboard buffer. You can type in up to 85 characters, sit back, and drink your coffee while the code dribbles out. Additional memory could be added—up to 65 thousand characters, and that's what I call a big buffer. With only a little more memory, canned messages could be inserted and held for a later call-up (before power is turned off)—CQs, tests, QTHs, for example. As the program is right now, only what is typed will be sent out.

Almost any crystal between 1 and 4 MHz can be used, so the old junk box can be used for some of the components. Look in an old color TV set—the 3.58-MHz crystal works well, too. Reset is provided by the resistor-capacitor combination automatically upon power-up conditions. If for any reason the computer does something strange or just quits, turn the power off and back on again. This will reset everything and start over. A note about the keyboard: The program is set up to recognize any keyboard that provides the ASCII code

```
E0F7 48                 CPWS   ASLA
E0F8 2A 08 E102                BPL    CPCS      CHECK FOR SPACE
E0FA C6 EF         A           LDAB   #$EF
E0FC D7 05         A           STAB   RESMSK    WORD SPACE RESET
E0FE C6 08         A           LDAB   #8        WORD SPACE BEING SENT
E100 20 06 E108                BRA    CKCNT

E102 C6 F7         A CPCS      LDAB   #$F7
E104 D7 05         A           STAB   RESMSK
E106 C6 04         A           LDAB   #4
E108 D1 04         A CKCNT     CMPB   COUNT
E10A 27 06 E112                BEQ    CK1
E10C 7C 0004       A           INC    COUNT
E10F 7E E084       A RETRN     JMP    POLL

E112 7F 0004       A CK1       CLR    COUNT
E115 96 0A         A           LDAA   COSTA
E117 94 05         A           ANDA   RESMSK
E119 97 0A         A           STAA   COSTA
E11B 26 F2 E10F                BNE    RETRN
E11D 20 14 E133                BRA    GNEL

                     ***SEND OUT A ZERO***
E11F B6 8006       A SOZERO    LDAA   PIA1BD
E122 84 FE         A           ANDA   #$FE
E124 B7 8006       A           STAA   PIA1BD
E127 39                        RTS
E128 20 64 E18E COMMR1         BRA    COMMR

                     ***SEND OUT A ONE***
E12A B6 8006       A SOONE     LDAA   PIA1BD
E12D 8A 01         A           ORAA   #1        SET BIT ZERO
E12F B7 8006       A           STAA   PIA1BD
E132 39                        RTS

                     ***GET NEW ELEMENT FOR OUTPUT***
E133 96 6D         A GNEL      LDAA   BUFTOP
E135 81 80         A           CMPA   #$80      TEST FOR CHAR END
E137 26 26 E15F                BNE    GNEL1
E139 86 08         A           LDAA   #$08      SEND OUT CHAR. SPACE
E13B 97 0A         A           STAA   COSTA
E13D 8D E0 E11F                BSR    SOZERO
E13F DE 02         A           LDX    SAVEX
E141 8C 006D       A           CPX    #BUFTOP
E144 26 09 E14F                BNE    GNELR
E146 96 06         A           LDAA   BUFLAG    BUFFER EMPTY
E148 84 7F         A           ANDA   #$7F
E14A 97 06         A           STAA   BUFLAG    RESET FLAG
E14C 7E E084       A           JMP    POLL
                     *
E14F 8D 2D E17E GNELR          BSR    MOVUP
E151 96 6D         A           LDAA   BUFTOP
E153 81 21         A           CMPA   #$21      LOOK FOR SPACE
E155 27 1A E171                BEQ    SWS
E157 7D 000A       A           TST    COSTA
E15A 27 03 E15F                BEQ    GNEL1
E15C 7E E084       A           JMP    POLL
                     *
E15F 78 006D       A GNEL1     ASL    BUFTOP
E162 25 04 E168                BCS    SODAH     SEND OUT DAH
E164 86 A0         A           LDAA   #$A0      SEND OUT DIT & ELEMENT SPA
E166 20 02 E16A                BRA    SOEL
                     *
E168 86 60         A SODAH     LDAA   #$60      SEND OUT DAH &ELEMENT SPAC
E16A 97 0A         A SOEL      STAA   COSTA     SET STATUS
E16C 8D BC E12A                BSR    SOONE
E16E 7E E0B4       A           JMP    POLL
                     *
E171 86 80         A SWS       LDAA   #$80
E173 97 6D         A           STAA   BUFTOP
E175 86 10         A           LDAA   #$10
E177 97 0A         A           STAA   COSTA     SEND OUT WORD SPACE
E179 8D A4 E11F                BSR    SOZERO
E17B 7E E0B4       A           JMP    POLL
                     *
E17E CE 0055       A MOVUP     LDX    #BUFTOP-BUFBOT
E181 A6 17         A MOV1      LDAA   BUFBOT-1,X
E183 A7 18         A           STAA   BUFBOT,X
E185 09                        DEX
E186 26 F9 E181                BNE    MOV1
E188 DE 02         A           LDX    SAVEX     UPDATE INPUT POINTER
E18A 08                        INX
E18B DF 02         A           STX    SAVEX
E18D 39                MOVRT   RTS

                     ***COMM REC INTERRUPT ROUTINE***
E18E DE 02         A COMMR     LDX    SAVEX
E190 B6 8020       A           LDAA   PIA2AD    GET KB DATA
E193 8C 0018       A           CPX    #BUFBOT
E196 27 49 E1E1                BEQ    BUFFUL    MEMORY BUFFER FULL
E198 81 5A         A           CMPA   #$5A
E19A 22 49 E1E5                BHI    BADCH     ILLEGAL CHAR
E19C 97 01         A STM       STAA   CVCX+1
E19E DE 00         A           LDX    CVCX
E1A0 E6 00         A           LDAB   0,X       DATA CONVERTED TO MORSE CO
E1A2 27 41 E1E5                BEQ    BADCH     KICK OUT BAD CHAR'S
E1A4 DE 02         A           LDX    SAVEX
E1A6 09                        DEX
E1A7 E7 00         A           STAB   0,X       STORE DATA IN BUFFER
E1A9 DF 02         A           STX    SAVEX     MOVE POINTER DOWN
E1AB D6 06         A           LDAB   BUFLAG
E1AD CA 80         A           ORAB   #$80      SET FLAG BIT 7
E1AF D7 06         A           STAB   BUFLAG
E1B1 8D 03 E1B6 PRNT           BSR    OUTCH     PRINT CHARACTER
E1B3 7E E0B4       A           JMP    POLL
                     ***PRINT CHARACTER ROUTINE***
E1B6 36                OUTCH   PSHA             SAVE LSB CHARACTER
E1B7 C6 C0         A           LDAB   #$C0
E1B9 CE 0009       A           LDX    #TMPSVE+2
E1BC A6 00         A           LDAA   0,X
E1BE 8D 10 E1D0 OUTCH1         BSR    STROBE
E1C0 09                        DEX
E1C1 A6 00         A           LDAA   0,X
E1C3 A7 01         A           STAA   1,X
E1C5 C0 40         A           SUBB   #$40
E1C7 26 F5 E1BE                BNE    OUTCH1
E1C9 32                        PULA
E1CA 84 3F         A           ANDA   #$3F      STRIPS BIT 6 AND 7
E1CC 97 07         A           STAA   TMPSVE
E1CE 20 00 E1D0                BRA    STROBE
E1D0 1B                STROBE  ABA
E1D1 B7 8022       A           STAA   PIA2BD    WRITE INTO APPROPRIATE DIG
E1D4 86 34         A           LDAA   #$34      BIT 3 = 0
E1D6 B7 8023       A           STAA   PIA2BC    PULSES CB2 LOW
E1D9 86 3C         A           LDAA   #$3C      BIT 3 = 1
E1DB B7 8023       A           STAA   PIA2BC    AND BACK AGAIN FOR
E1DE 39                        RTS              WRITE PULSE
E1DF 20 AD E18E COMMR2         BRA    COMMR
                     *
E1E1 86 2A         A BUFFUL    LDAA   #'*       PRINT '*' FOR BUFFER FULL
E1E3 20 CC E1B1                BRA    PRNT
                     *
                     *
E1E5 86 23         A BADCH     LDAA   #$23      PRINT '#' FOR BAD CHAR
E1E7 20 C8 E1B1                BRA    PRNT

                     ***START OF RECEIVE ROUTINE***
E1E9 B6 8006       A REC       LDAA   PIA1BD
E1EC 8A 04         A           ORAA   #4        REC LED ON
E1EE 84 FD         A           ANDA   #$FD      XMIT LED OFF
E1F0 B7 8006       A           STAA   PIA1BD
E1F3 86 FF         A KEYUP     LDAA   #$FF      RESET INTERVAL COUNTER
E1F5 81 FE         A KULOOP    CMPA   #$FE      INTERVAL COUNTER AT MAX?
E1F7 27 03 E1FC                BEQ    NOINC     DO NOT INCR IF MAX
E1F9 4C                        INCA             INCREMENT INTERVAL COUNTER
E1FA 20 0A E206                BRA    KUCONT

E1FC 97 14         A NOINC     STAA   KUTIM     SAVE KU INTERVAL TIME
E1FE D6 0F         A           LDAB   RCHAR     GET RCVD CHAR
E200 C1 01         A           CMPB   #1        ANYTHING THERE?
E202 27 02 E206                BEQ    KUCONT
E204 8D 4C E252                BSR    COMPKU
E206 F6 8004       A KUCONT    LDAB   PIA1AD    CHECK INPUT
E209 2B 09 E214                BMI    KD        BRANCH IF KEYDOWN
E20B 8D 36 E243                BSR    TIMER     TIME DELAY

E20D 7D 8021       A           TST    PIA2AC    TESTS BIT 7 TO SEE IF DATA
E210 2B CD E1DF                BMI    COMMR2    IF DATA THN XMIT
E212 20 E1 E1F5                BRA    KULOOP
E214 97 14         A KD        STAA   KUTIM     SAVE KU INTERVAL TIME
E216 8D 3A E252                BSR    COMPKU
E218 86 FF         A KEYDWN    LDAA   #$FF      RESET INTERVAL TIMER
E21A 81 FE         A KDLOOP    CMPA   #$FE      INTERVAL COUNTER AT MAX?
E21C 27 01 E21F                BEQ    MAXKD     DO NOT INCR IF MAX
E21E 4C                        INCA             INCREMENT INTERVAL COUNTER
E21F F6 8004       A MAXKD     LDAB   PIA1AD    CHECK INPUT
E222 2A 04 E228                BPL    KU        BRANCH IF KEYUP
E224 8D 1D E243                BSR    TIMER
E226 20 F2 E21A                BRA    KDLOOP

E228 97 13         A KU        STAA   KDTIM     SAVE KD INTERVAL TIME
E22A 81 04         A           CMPA   #$4       KD INTERVAL TIME TOO LOW?
E22C 24 05 E233                BCC    CKHI      BRANCH IF NOT TOO LOW
E22E 96 0E         A           LDAA   SPEEDK
E230 44                        LSRA             DIVIDE SPEED CONSTANT BY 2
E231 20 07 E23A                BRA    UNZERO

E233 81 7F         A CKHI      CMPA   #$7F      KD INTERVAL TIME TOO HIGH?
E235 25 07 E23E                BCS    CMPTKD    BRANCH IF OK
E237 96 0E         A           LDAA   SPEEDK
E239 48                        ASLA             MULTIPLY SPEED CONSTANT BY
```

Continued

set and a negative-going strobe. There are many available from wholesale houses for $20.00 and up.

Key Characters Available

The computer will generate all of the Morse characters plus some special function keys which can be generated by the use of the CNTRL key. A list of them follows:

Control A	–	KN
B	–	BK
C	–	AR
D	–	SK
F	–	SN
H	–	ERROR (8 dots)
=	–	BT
ESC	–	AS
SPACE	–	SPACE

A "space" will insert a space in the buffer to be transmitted along with the code, thereby making perfect Morse every time. While typing, if you should reach the top of the buffer, the character you try to enter will be displayed as an *, meaning that it did not get entered and you should reenter it after one character has been sent.

Any illegal character typed will appear as a # and will not get sent. Any received character which the computer cannot figure out—like run-together characters—will be displayed as an __ and an error (8 dots) will be an @.

Parts Procurement

All parts (with the exception of the DL-1416) can be obtained from your local Motorola distributor, and the 1416 comes from a Litronix distributor. All of the parts should tally to

```
00363A E23A 8A 01     A UNZERO ORAA   #1        ASCERTAIN SPEED CONSTANT I
00364A E23C 97 0E     A        STAA   SPEEDK    NOT SET TO ZERO
00365A E23E 8D 31 E271 CMPTKD BSR     COMPKD
00366A E240 7E E1F3   A        JMP    KEYUP

00368                   ***SUBROUTINE TO CREATE TIME DELAY***
00369A E243 37          TIMER  PSHB             SAVE B
00370A E244 36                 PSHA             SAVE A
00371A E245 D6 0E     A        LDAB   SPEEDK
00372A E247 86 40     A DELOP2 LDAA   #$40
00373A E249 4A          DELOOP DECA
00374A E24A 26 FD E249         BNE    DELOOP
00375A E24C 5A                 DECB
00376A E24D 26 F8 E247         BNE    DELOP2
00377A E24F 32                 PULA             RESTORE A
00378A E250 33                 PULB             RESTORE B
00379A E251 39                 RTS

00381                   ***SUBROUTINE TO COMPUTE KU***
00382A E252 91 11     A COMPKU CMPA   TQLDAT
00383A E254 25 1A E270         BCS    MOREL     BRANCH IF KUTIM ` TQLDAT
00384A E256 96 0F     A        LDAA   RCHAR     GET CHAR BEING RECEIVED
00385A E258 81 01     A        CMPA   #1
00386A E25A 27 09 E265         BEQ    CKFSP
00387A E25C 8D 4A E2A8         BSR    GAFT      GET ASCII FROM TABLE
00388A E25E BD E1B6   A        JSR    OUTCH     PRINT CHARACTER IN ACCA
00389A E261 86 01     A        LDAA   #1
00390A E263 97 0F     A        STAA   RCHAR     READY FOR NEW CHAR
00391A E265 96 12     A CKFSP  LDAA   TLDAT     GET TWICE LAST DASH TIME
00392A E267 91 14     A        CMPA   KUTIM     COMPARE WITH KU INTERVAL
00393A E269 24 05 E270         BCC    MOREL     BRANCH IF TLDAT "= KU INTE
00394A E26B 86 20     A        LDAA   #$20      ASCII SPACE
00395A E26D BD E1B6   A        JSR    OUTCH     PRINT SPACE
00396A E270 39          MOREL  RTS

00398                   ***SUBROUTINE TO COMPUTE KD***
00399A E271 96 0D     A COMPKD LDAA   TLETIM    GET TWICE LAST ELEM TIME
00400A E273 91 13     A        CMPA   KDTIM     COMPARE WITH KD INTERVAL
00401A E275 25 10 E287         BCS    DASHEL    BRANCH IF TLETIM ` KD INTE
00402A E277 96 0C     A        LDAA   HLETIM    GET HALF LAST ELEM TIME
00403A E279 91 13     A        CMPA   KDTIM     COMPARE WITH KD INTERVAL
00404A E27B 24 04 E281         BCC    DOTEL     BRANCH IF HLETIM "= KD INT
00405A E27D 96 0B     A        LDAA   LETYPE    CHECK LAST ELEMENT TYPE
00406A E27F 26 06 E287         BNE    DASHEL    BRANCH IF LAST ELEM WAS DA
00407A E281 7F 000B   A DOTEL  CLR    LETYPE    MAKE LAST ELEM TYPE=00=DOT
00408A E284 0C                 CLC
00409A E285 20 14 E29B         BRA    ADDEL
00410A E287 7C 000B   A DASHEL INC    LETYPE    MAKE LAST ELEM TYPE=DASH
00411A E28A 96 13     A        LDAA   KDTIM     GET KD INTERVAL
00412A E28C 97 10     A        STAA   LDATIM    STORE IN LAST DASH TIME
00413A E28E 16                 TAB              SAVE IN ACCUM B
00414A E28F 44                 LSRA             DIVIDE KD INTERVAL BY 2
00415A E290 97 11     A        STAA   TQLDAT    SAVE 1/2 KD
00416A E292 44                 LSRA             DIVIDE 1/2 KD BY 2
00417A E293 9B 11     A        ADDA   TQLDAT    ADD 1/2 TO 1/4 KD INTERVAL
00418A E295 97 11     A        STAA   TQLDAT    STORE RESULT
00419A E297 58                 ASLB             MULTIPLY KD INTERVAL BY 2
00420A E298 D7 12     A        STAB   TLDAT     STORE RESULT
00421A E29A 0D                 SEC
00422A E29B 79 000F   A ADDEL  ROL    RCHAR     ADD NEW ELEM TO CHARACTER
00423A E29E 96 13     A        LDAA   KDTIM     GET KD INTERVAL
00424A E2A0 16                 TAB              SAVE IN ACCUM B
00425A E2A1 44                 LSRA             DIVIDE KD BY 2
00426A E2A2 97 0C     A        STAA   HLETIM    STORE 1/2 KD INTERVAL
00427A E2A4 58                 ASLB             MULTIPLY KD BY 2
00428A E2A5 D7 0D     A        STAB   TLETIM    STORE TWICE KD INTERVAL
00429A E2A7 39                 RTS

00431                   ***SUBR. TO GET ASCII CHAR FROM CODE TABLE***
00432A E2A8 0D          GAFT   SEC              CHANGE FORMAT OF RCHAR.
00433A E2A9 49                 ROLA
00434A E2AA 48          GAFT1  ASLA
00435A E2AB 24 FD E2AA         BCC    GAFT1
00436A E2AD CE E05A   A        LDX    #RESRT-1
00437A E2B0 A1 00     A STAB1  CMPA   0,X
00438A E2B2 27 09 E2BD         BEQ    TABM      FOUND MATCH
00439A E2B4 09                 DEX
00440A E2B5 8C E021   A        CPX    #RTAB+1   END OF TABLE?
00441A E2B8 26 F6 E2B0         BNE    STAB1     NO!
00442A E2BA 86 5F     A        LDAA   #'_       RETURN "_" FOR NO MATCH
00443A E2BC 39                 RTS
00444                   *
00445A E2BD DF 16     A TABM   STX    RECX
00446A E2BF 96 17     A        LDAA   RECX+1    X(LOW) TO ACCA.
00447A E2C1 39                 RTS

00449                   ***INITIALIZATION AND RESTART VECTORS***

00451A E3F8                    ORG    CODE+$3F8
00452A E3F8    E0B4   A        FDB    POLL      IRQ
00453A E3FA    E05B   A        FDB    RESRT     SWI
00454A E3FC    E05B   A        FDB    RESRT     NMI
00455A E3FE    E05B   A        FDB    RESRT     RESET
00456                          END
TOTAL ERRORS 00000
```

```
E29B ADDEL  00409 00422*
E1E5 BADCH  00273 00277 00315*
0018 BUFBOT 00059*00257 00258 00259 00270
E1E1 BUFFUL 00271 00311*
0006 BUFLAG 00042*00150 00227 00229 00282 00284
006D BUFTOP 00060*00115 00119 00218 00225 00233 00240 00251 00257
E0DC CES1   00165*
E102 CFCS   00183 00189*
E0DF CFDAH  00162 00167*
E0EA CFES   00168 00174*
E0F7 CFWS   00175 00182*
0015 CHCTR  00057*
E112 CK1    00193 00197*
E108 CKCNT  00166 00172 00180 00187 00192*
E265 CKFSP  00386 00391*
E233 CKHI   00354 00359*
E23E CMPTKD 00360 00365*
E000 CODE   00081*00117 00451
E18E COMMR  00209 00268*00309
E128 COMMR1 00149 00209*
E1DF COMMR2 00309*00339
E271 COMPKD 00365 00399*
E252 COMPKU 00333 00342 00382*
000A COSTA  00044*00161 00198 00200 00222 00236 00246 00253
0004 COUNT  00040*00192 00194 00197
0000 CVCX   00038*00118 00274 00275
E287 DASHEL 00401 00406 00410*
E249 DELOOP 00373*00374
E247 DELOP2 00372*00376
E281 DOTEL  00404 00407*
E0B1 EXEC   00142*00143
E2A8 GAFT   00387 00432*
E2AA GAFT1  00434*00435
E133 GNEL   00202 00218*
E15F GNEL1  00220 00237 00240*
E14F GNELR  00226 00232*
000C HLETIM 00048*00402 00426
FFF8 IRQVEC 00077*00122
E214 KD     00335 00341*
E21A KDLOOP 00344*00350
0013 KDTIM  00055*00352 00400 00403 00411 00423
E218 KEYDWN 00343*
E1F3 KEYUP  00323*00366
E228 KU     00348 00352*
E206 KUCONT 00327 00332 00334*
E1F5 KULOOP 00324*00340
0014 KUTIM  00056*00329 00341 00392
E05E L1     00106*00108
0010 LDATIM 00052*00412
000B LETYPE 00047*00405 00407 00410
E21F MAXKD  00345 00347*
E270 MOREL  00383 00393 00396*
E181 MOV1   00258*00261
E18D MOVRT  00265*
E17E MOVUP  00232 00257*
E1FC NOINC  00325 00329*
E0C6 NOTRCV 00151 00153*
E1B6 OUTCH  00285 00288*00388 00395
E1BE OUTCH1 00292*00297
8005 PIA1AC 00065*00146
8004 PIA1AD 00064*00123 00156 00334 00347
8007 PIA1BC 00067*
8006 PIA1BD 00066*00157 00160 00205 00207 00212 00214 00319 00322
8021 PIA2AC 00072*00135 00148 00338
8020 PIA2AD 00071*00133 00269
8023 PIA2BC 00074*00139 00305 00307
8022 PIA2BD 00073*00137 00303
E0B4 POLL   00121 00146*00195 00230 00238 00248 00255 00286 00452
E0C7 POLL2  00147 00156*
E1B1 PRNT   00285*00312 00316
000F RCHAR  00051*00330 00384 00390 00422
E1E9 REC    00152 00319*
0016 RECX   00058*00445 00446
0005 RESMSK 00041*00164 00170 00177 00185 00190 00199
E05B RESRT  00105*00436 00453 00454 00455
E10F RETRN  00195*00201
E020 RTAB   00088*00440
0002 SAVEX  00039*00120 00224 00262 00264 00268 00278 00281
E168 SODAH  00241 00245*
E16A SOEL   00243 00246*
E12A SOONE  00212*00247
E11F SOZERO 00178 00205*00223 00254
000E SPEEDK 00050*00113 00355 00361 00364 00371
E2B0 STAB1  00437*00441
007F STACK  00076*00140
E19C STM    00274*
E1D0 STROBE 00292 00301 00302*
E171 SWS    00235 00250*
E2BD TABM   00438 00445*
E243 TIMER  00336 00349 00369*
0012 TLDAT  00054*00391 00420
000D TLETIM 00049*00399 00428
0007 TMPSVE 00043*00110 00111 00290 00300
0011 TQLDAT 00053*00382 00415 00417 00418
E23A UNZERO 00357 00363*
```

less than $80-$90, including the display, which is $30 in quantities of one.

Operation

Operation of the reader-talker is quite simple. Hook up the required power supplies, the cords to the speaker and key jacks, and go to town. The easiest way to tune in a signal is to use the on-board headphone jack and set the selectivity to 40 Hz. When you hear the signal, watch the LED, and when it starts to blink at the incoming CW rate, switch the filter to the processed mode and tune for the cleanest signal. An RIT control is almost a must as the input tuning is quite sharp, and if you tune the other guy for the best signal each time he gives it back to you, you could walk right up or down the band!

The only drawback I have found is that when copying at fast speeds, the display will run words together, since most operators do not leave enough space between them. I understand that most of the keyboard keyers do not have a space key on them, and this will explain some of the problems. The others are self-explanatory. As I said earlier, the lack of more digits on the board is not a hindrance, and after you use it a while, you'll agree. Whether you use a PC board or wire-wrap this project, it will be a great addition to any ham shack and do a lot for cleaning up the airways. ■