Designing the Lakeland Repeater (WR4 AWJ) Control and Identify Program

By Bill Hunsicker (K4DF/W5HKM)

An old college friend, George P. Burdell, dropped by to see me not too long ago. After trading a few amenities, we began discussing computer applications.

"Bill, I had an idea about using a microcomputer to run the Lakeland Amateur Radio Repeater."

"I knew it! OK, I can give you some of what you want, but just what are we setting out to accomplish? In other words, what are our goals? Here, let me show you what I mean." I then wrote 9 goals on the blackboard:

GOALS

1. Recognize input signals. (COR and others later on)
2. Control the repeater transmitter.
3. Keep accurate time. (transparent to current operations)
4. Perform ID on a time scheduled basis.
5. Maintain and update internal timers.
6. Provide windbag, tailgate and transmitter time-out control.
7. Provide tone-modulated code. (also transparent)
8. Provide a "Kerchunk" filter.
9. Provide for future growth. (Phone-Patch, etc.)

I said, "The next thing we should do is to define the steps that need to be accomplished."

George interrupted me. "Then we need to develop a flow chart, write the program, develop the hardware. . ." Our list grew to a total of eight items.

STEPS TO ACCOMPLISH

1. Develop list of goals.
2. Develop Flowchart.

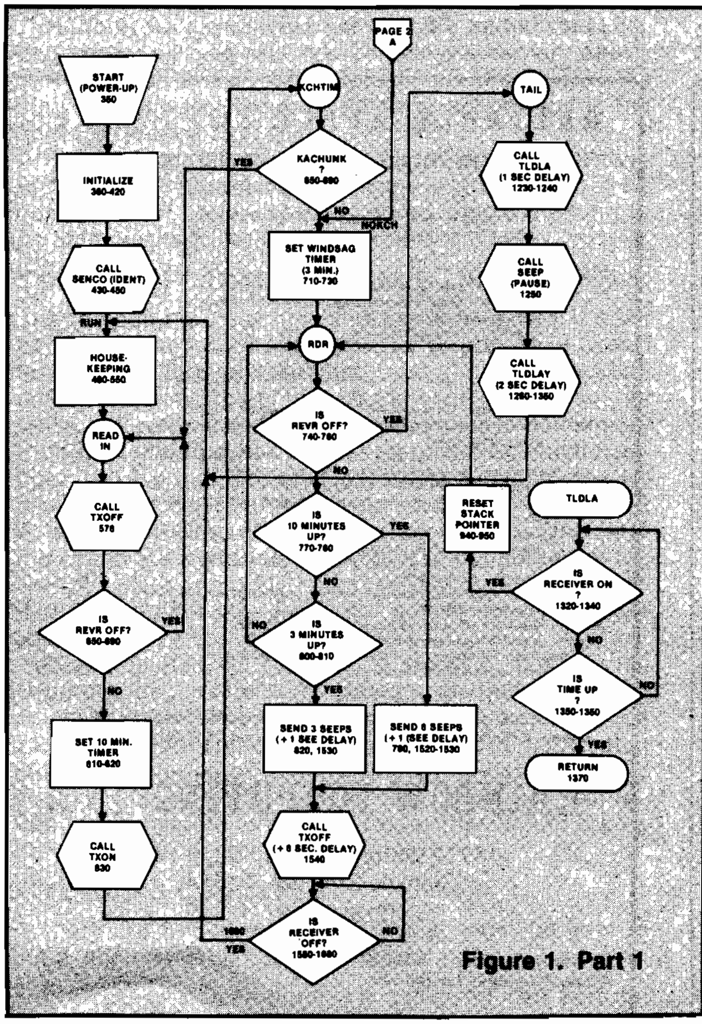


Figure 1. Part 1

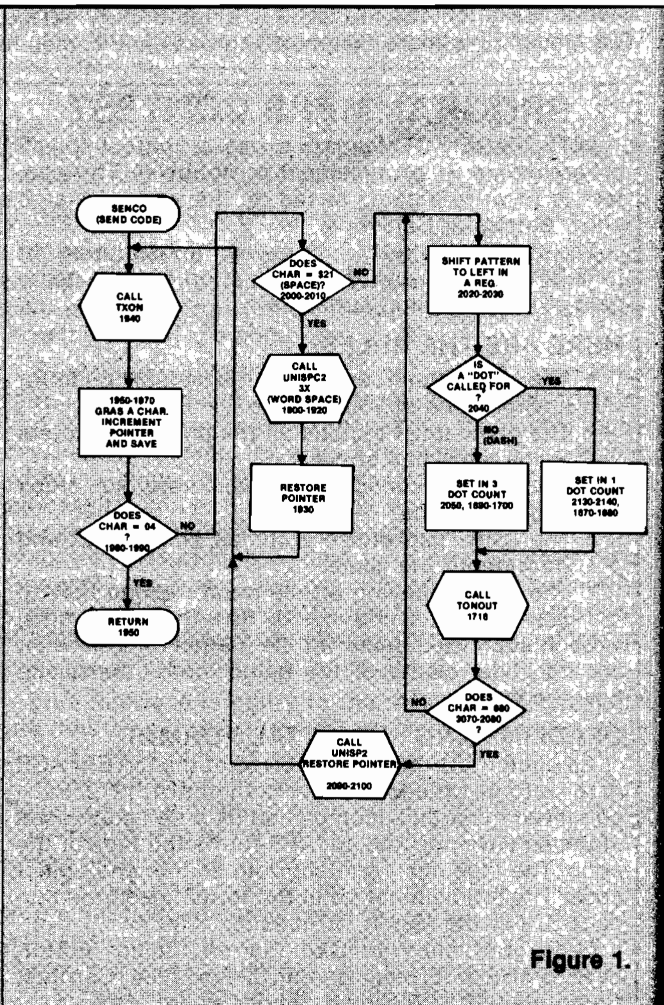


Figure 1.

3. Write the program and develop hardware circuit.
4. Simulate and debug the software and hardware.
5. Convert the software to firmware (EPROM).
6. Build hardware.
7. Connect to Repeater.
8. Plan enhancements.

"Now that we have come this far, George, we have reached a milestone. We have done the preliminary planning. I anticipated a request like yours; steps 1 through 5 of the 'steps to accomplish' have already been done!"

"Here is a copy of the assembled program listing (Program 1), and the schematic (Figure 1). Now then, if you will take the program to your friendly microcomputer store, have them program a 2708 EPROM for you and then build the circuit I gave you, you are almost home free."

"Hey, that's fantastic, Bill. It's more than I had even hoped for. But I'd like to know how this thing works. Suppose it sent the code too fast or too slow, or that the call letters have to be changed? What about. . ."

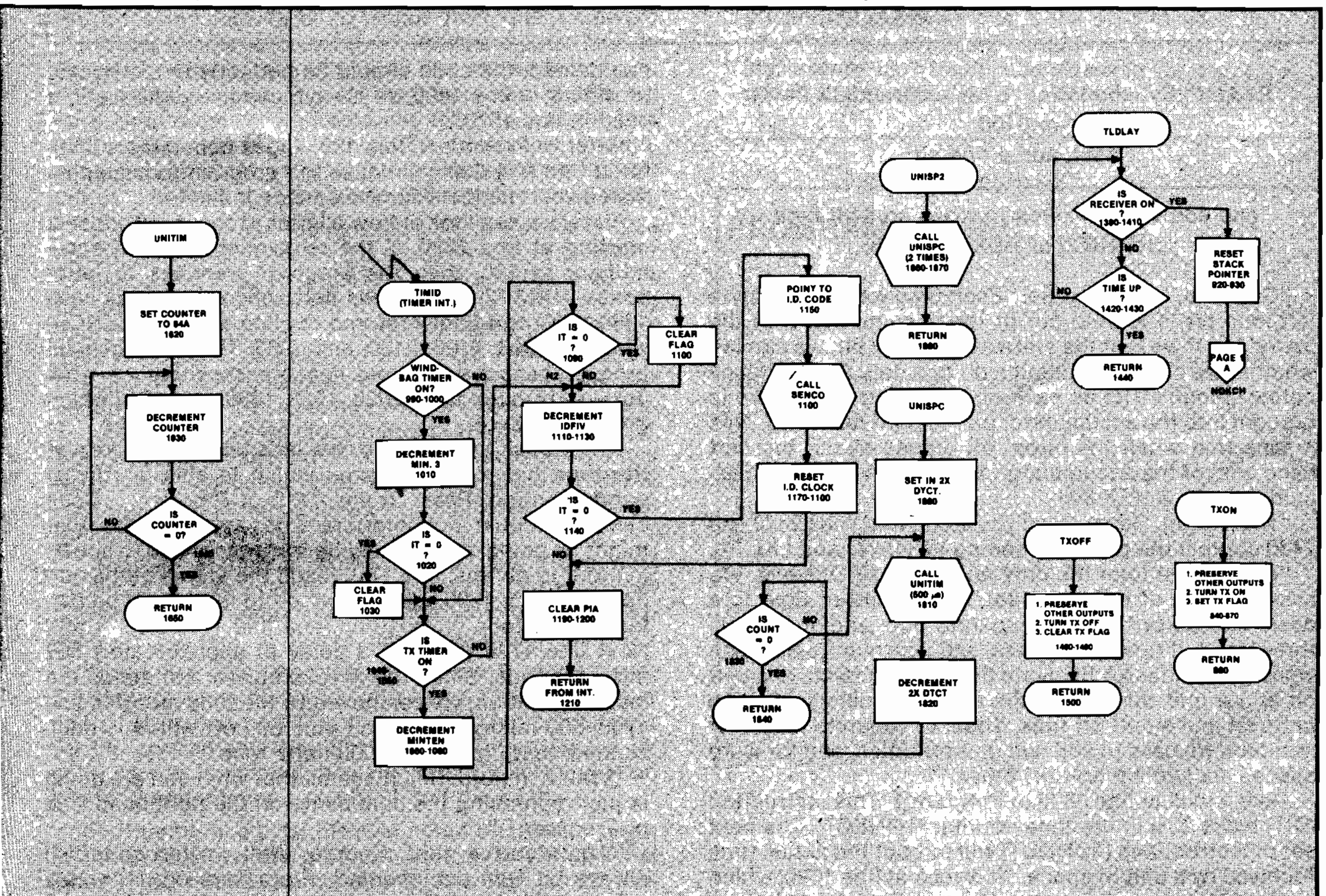
"Thought you'd never ask, George. The main program starts at the label START where system initialization is performed. Once the system has been initialized, the computer will fall into a loop at label READIN (lines 570-600) where the transmitter will be turned off. The system will then wait for an indication that the receiver has detected a carrier (HEX bit 01). After a carrier has been detected, the computer will proceed to set the ten minute timer (maximum transmitter on-time) and to turn the transmitter on. While the transmitter is keyed, the operational loop at RDR (lines 740-810) is in control of the system.

"There are two main support routines in the program. One of the main subroutines is TIMID, which is the time keeping routine activated by the hardware clock ticks. The other subroutine is SENCO, the code sending routine.

"First off, let's see how the code is sent. Before SENCO is called, the index register (line 1150) is pointed at the code buffer IDENT. The transmitter is turned on at line 1960, the first byte is loaded into the A accumulator (line 1970), and then the index register is incremented by one and saved. Now we check to see if the A accumulator contains the terminator for sending code (end of transmission is a hexadecimal 04). If it does, we go back to the caller; else we check to see if it is a space between words (HEX 21).

"If it was a word space we do a WRDSPC; if not, we do an arithmetic shift left, push the A accumulator on the stack (to save it) and check to see if there is a carry. If there is a carry, we transmit DAH (dash); if not, we transmit a DIT (dot). Next we retrieve the value we saved in the stack and put it back into the A accumulator. If the character was a HEX 80, we transmit a letter space and re-enter the routine at SENCO. If the character retrieved was not a HEX 80, we do another arithmetic shift left (ASL), and so on.

"Let me show you how to construct a code message. You see, we let zeros be dits and the ones be dahs, all to be followed by a one."



Part 2

Figure 1. Part 3

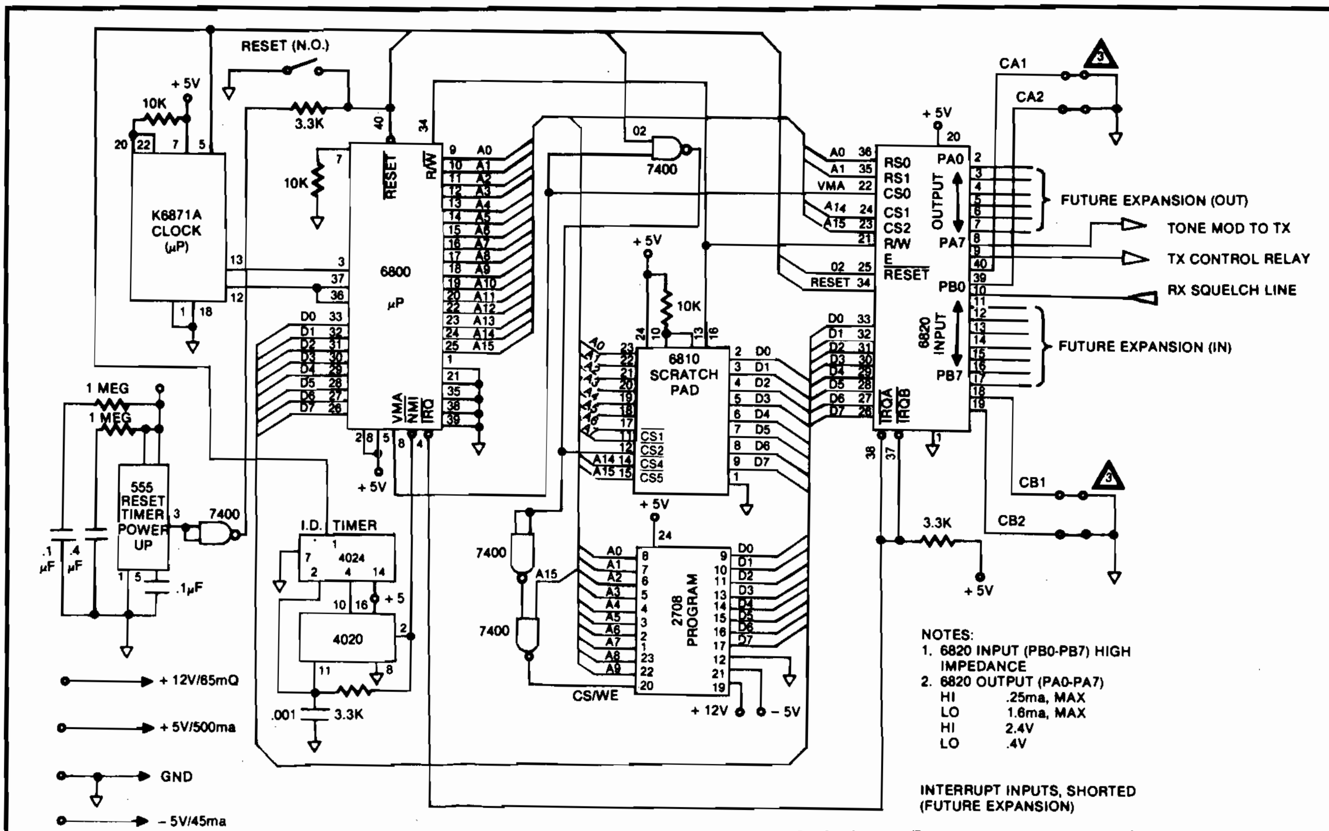


Figure 3. Computer Control for the Lakeland Repeater — WR4AWJ.

EXAMPLE: the letter "L" is a didahdidit (.—..)
or 0100
next, add a one bit:
or 01001
and add enough zeros to make an 8-bit byte:
01001000 or HEX 48 ($48)

"Oh, I see," said George. "Each time we shift the byte left, one bit is shifted out of the accumulator into the 'carry' bit. If we test the state of the carry bit, we can tell whether we should transmit a dit or a dah. But the fascinating thing is that as each bit is shifted left, a zero is brought into the accumulator on the right hand end. This means that when the 'stop' bit arrives at the top (left hand end) of the accumulator, the actual value of the accumulator is a HEX 80."

"That's right, George, and you'll find that lines 2420 and 2430 contain the data for sending WR4AWJ and LKL FL.

"How do we form the dots and dashes? Observe. We assign a unit of time to a dot, three units of time to a dash, one unit of time to separate dots and/or dashes, three units of time characters, and seven units between words. Consider AA EE. Note that a dit consists of a unit on and a unit off time, while a dash consists of three units on and a unit off time. Thus, to separate letters or numbers we have to add two unit off times, and for word spaces we add six unit off times."

Take a look at line 1670 in Program 1. This instruction (DIT) loads $2B in the tone counter (TONECT). The next set of instructions (TONOUT) will put out the actual tone followed by a unit of space. Now look at line 1690, which loads $81 in the tone counter and then branches to TONOUT to put out the actual tone. The ratio of $2B to $81 is 1 to 3 (43 to 129 decimal). As long as this ratio is maintained, the code should be perfectly proportioned, providing in line 1800 we always use twice the number used in line 1670.

Refer to Program 1, line 150. These constants substituted into line 230 will cause the program to assemble the correct numbers for the speeds indicated.

"Now we can look at how a letter space is introduced, starting at line 2090. If the remainder of the A accumulator is $80, then the program branches to letter space (UNISP2) which calls unit space twice (see lines 1880-1890) and finally returns to code sending (SENCO) via lines 1900, 2120, 2130, and 1960. The same general process happens for word space except that we work our way through lines 2030, 1920, 1930, and 1940 which gives us 3*2=6.

"That wasn't so hard, so now let's take a close look at the tone generation scheme which is also worthy of some mention. Refer to line 1710 of Program 1. Code sending should not interfere with anything else that might be going on; it should be transparent, so to speak. In order to do this, line 1710 reads the status of the output lines, line 1720 adds an 'on' bit to the output bit pattern in the tone bit position, and line 1730 puts it out. Next line 1740 goes for a delay of approximately 500 microseconds. Since the A accumulator still holds the last information, we can now turn the tone bit off in line 1760. 500 microseconds on plus 500 off will be 1 millisecond, just right for a 1000 hertz tone."

"Wait a minute, Bill, it seems that what you are doing is just rephrasing the comments on the right hand side of Program 1."

"Again you're right, George. Well written code will always be well documented. From here on, let's only refer generally to Program 1, and you can read it for yourself. Let's examine the UNISPC delay loop and the 500 microsecond delay loop.

"Line 1820 seems to infer that we need a space twice as long as a dit. Actually, a dit is turned on for 500 microseconds and then off for 500 more. Therefore, to equal the length of a dit, we must have two 500 microsecond delays for each dit count (DTCT). Thus, we use lines 1820 through 1860 to obtain a delay equal to a dit time.

"The 500 microsecond delay loop, on the other hand, is a little more sticky and is the basis of most of our delays. The crystal clock assumed is approximately 1.797 Mhz which makes each machine cycle approximately 1.113 microseconds long. If we trace the program from line 1620 through lines 1630, 1640, 1650 and back, we will have consumed 445 machine cycles. 445*1.11 microseconds per machine cycle is approximately 495 microseconds."

"Bill, if all the timing is developed by your 500 microsecond delay loop, why on the schematic (Figure 1) do you have a 4024 and 4020 identified as a ID TIMER? It seems to be attached to the 6800 NMI input. Isn't that being used as a 'timer' of sorts?"

"There are two general types of timing employed by the program. One we have just examined is related to the system master crystal controlled clock to generate the 500 microsecond delays. There is another form of timing that depends on a hardware clock. The reason is, if we tied the CPU up in being a software clock, it would be hard-pressed to do anything else. Note that the 500 microsecond delays are when it is doing a specific chore (sending code) and almost everything else stays in a 'status-quo' condition. Almost everything, that is. We do allow interrupts to occur from the hardware clock. My simulations using the hardware clock tied to the non-maskable interrupt (NMI) line showed the clock ticks to be unnoticeable. If they were noticeable, we have an alternative; we could use the interrupt request (IRQ) line which is maskable. The difference is negligible, and at worst we would only lose several seconds out of every five minutes while sending the ident code.

"If we want to use the hardware clock (4020 and 4024 dividers) for IRQ operation, it probably would be better to open a link (CB2 to GND) and configure the 6820 PIA for interrupts through the IRQB line. In this case, we must do a 'read' in the TIMID routine. See Figure 1. Line 2490 will have to be changed to a "FDB TIMID" instruction and line 2510 changed to "RMB 1". This now vectors us into the time clock routine via the interrupt on IRQB.

"With the hardware clock made of the 4024 and 4020, the closest to a 1 second time tick that could be obtained reasonably was .88 seconds. Thus lines 460, 610 and 710 contained calibrated numbers which provide the desired timings.

"Another approach would be to use presettable counters to give exactly one second or to use another 555 timer, free running, with an adjustment to put it exactly on one second pulses. (My simulation does this.)

"Since we are discussing different approaches, Motorola has just come out with a new CPU (which they call MPU). It is the 6802. It has the 6871A clock (except for the crystal) and the 6810 built in. This means a TV color crystal can probably be tied directly to two pins on the 6802 and eliminate the 6871 and the 6810.

"Furthermore, a 2704 EPROM could be used in place of the 2708 since the program uses only 427 decimal bytes. I do not recommend this, however, since this should only be the start of something big! Later additions to the program can do various neat things, occasionally calling on the already existing routines."

"How about phone patches, weather bulletins, off frequency reports, and so on?"

"Let me make an off-hand comment, George. This program has not been optimized, and there are places where the index register was used to permit 16 bit arithmetic to allow for the slower code speeds in case someone should want to lift the SENCO subroutine for some other code sending application. But for now, this is very convenient for us."□

ACKNOWLEDGEMENT

I would like to thank Brand-Rex, Teltronics Division for the use of their high speed printer which saved much time on this project.

ABOUT THE AUTHOR

Bill Hunsicker, K4DF/W5HKM, age 53, is a professional engineer #3544 Oklahoma. He received his amateur license at age 14, and graduated from Georgia Tech with BEE in 1951. He served in the Navy during World War II, has a private pilot's license, and a radio telephone license first class.

He took up personal computer programming a little over a year ago, and took a 3-day cram course from Motorola on the 6800. He has found that programming in software is just a step forward from what he had been doing in hardware decision making, except that trading hardware in for software opens a vast new field.

PROGRAM LISTING

```
00150                 * DTCT: $2B/20WPM, $38/15WPM, $68/10WPM
00160                        OPT    0
00170                 *
00180      4004       PIAD   EQU    $4004     OUTPUTS
00190      4005       PIAC   EQU    $4005
00200      4006       PIBD   EQU    $4006     INPUTS
00210      4007       PIBC   EQU    $4007
00220      007F       STK    EQU    $7F
00230      002B       DTCT   EQU    $2B       DIT COUNT
00240                 *
00250 0000                   ORG    $0
00260 0000 0001       MIN3   RMB    1
00270 0001 0002       MINTEN RMB    2
00280 0003 0002       IDFIV  RMB    2
00290 0005 0002       XTEMP  RMB    2
00300 0007 0002       TCNECT RMB    2
00310 0009 0001       TXFLG  RMB    1
00320 000A 0001       WINFLG RMB    1
00330                 *
00340 8000                   ORG    $8000
00350 8000 01         START  NOP
00360 8001 0F                SEI              SET INT MASK
00370 8002 CE FF1F           LDX    #$FF1F    ALL OUTPUTS
00380 8005 FF 4004           STX    PIAD      IN A SIDE
00390 8008 CE 001F           LDX    #$1F      ALL INPUTS
00400 800B FF 4006           STX    PIBD      IN B SIDE
00410 800E 8E 007F           LDS    #STK      SET STACK
00420 8011 CE 8190           LDX    #IDENT    IDENTIFY
00430 8014 BD 8145           JSR    SENCO     IDENT CODE
00440 8017 CE 819E           LDX    #RESET    CODE IDENT
00450 801A BD 8145           JSR    SENCO     SEND CODE
```

Continued on next page

```
801D CE 0108        LDX     #$108      5 MIN (.88)
8020 DF 03          STX     IDFIV      SET SOFT CLOCK
8022 8E 007F RUN    LDS     #STK       RESET STACK
8025 CE 4004        LDX     #PIAD      POINT AT PIA
8028 E6 02          LDA B   2,X        DUMMY READ
802A A6 00          LDA A   0,X        DUMMY READ
802C 7F 0009        CLR     TXFLG      CLEAR TX FLAG
802F 7F 000A        CLR     WINFLG     WINDBAG FLAG
8032 01             NOP                CDD-PARITY MASK
8033 0E             CLI                CLEAR INT. MASK
             *
8034 BD 80F5 READIN JSR     TXOFF      TURN TX OFF
8037 F6 4006        LDA B   PIBD       READ INPUTS
803A C4 01          AND B   #1         MASK OUT RX BIT
803C 27 F6          BEQ     READIN     RX OFF. DO AGN
803E CE 0210        LDX     #$210      10 MIN (.88)
8041 DF 01          STX     MINTEN     SET SOFT TIMER
8043 8D 2A          BSR     TXON       TURN TX ON
8045 CE FFFF        LDX     #$FFFF     FOR 1.5 SEC
8048 F6 4006 KCHTIM LDA B   PIBD       READ INPUTS
804B C4 01          AND B   #1         READ RX BIT
804D 27 E5          BEQ     READIN     DUMP TX
804F 09             DEX                COUNT DOWN
8050 26 F6          BNE     KCHTIM     TIMING
             *
8052 86 9D   NOKCH  LDA A   #$9D       3 MIN (.88)
8054 97 00          STA A   MIN3       WINDBAG TIMER
8056 7C 000A        INC     WINFLG     SET WIND FLAG
8059 F6 4006 RDR    LDA B   PIBD       READ INPUTS
805C C4 01          AND B   #1         READ RX BIT
805E 27 5A          BEQ     TAIL       NO. DELAY+BEEP
8060 DE 01          LDX     MINTEN     IS IT =0?
8062 26 03          BNE     *+5        NO. CONTINUE
8064 7E 80F1        JMP     WARN6      YES, SEND WARN
8067 7D 0000        TST     MIN3       IS IT =0?
806A 26 ED          BNE     RDR        NO. CONTINUE
806C 7E 80F4        JMP     WARN3      YES, SEND WARN
             *
806F C6 80   TXON   LDA B   #$80       TX BIT
8071 FA 4004        ORA B   PIAD       SAVE PRIOR
8074 F7 4004        STA B   PIAD       TURN TX ON
8077 7C 0009        INC     TXFLG      SET TX FLAG
807A 39             RTS
             *
             * JUMPED OUT OF LOOP
             *
807B 8E 007F RECON  LDS     #STK       RESET STK
807F 20 D2          BRA     NOKCH      NO WINDY
8080 8E 007F TLGATE LDS     #STK
8083 20 D4          BRA     RDR        TAILGATE RETURN
             *
             * NMI - 1 SEC PULSES
             *
8085 7D 000A TIMID  TST     WINFLG     WIND TIMER ON?
8088 27 08          BEQ     N1         NO, GET OUT
808A 7A 0000        DEC     MIN3       YES, DEC TIMER
808D 26 03          BNE     N1         TIMER=0?
808F 7F 000A        CLR     WINFLG     YES, CLR FLG
8092 7D 0009 N1     TST     TXFLG      TX TIMER ON?
```

```
810C 39             RTS
             *
810D CE 002B DIT    LDX     #DTCT      DIT COUNT
8110 20 03          BRA     *+5        STORE IT
8112 CE 0081 DAH    LDX     #3*DTCT    DAH COUNT
8115 DF 07          STX     TONECT     PUT IN COUNTER
8117 B6 4004 TONOUT LDA A   PIAD       GET STATUS
811A 8A 40          ORA A   #$40       TONE BIT ON
811C B7 4004        STA A   PIAD       PUT IT OUT
811F 8D E6          BSR     UNITIM     DELAY 500 MICRO SEC.
8121 84 BF          AND A   #$BF       TONE BIT OFF
8123 B7 4004        STA A   PIAD       PUT IT OUT
8126 8D DF          BSR     UNITIM     DELAY 500 MICROSEC
8128 DE 07          LDX     TONECT     GET COUNT
812A 09             DEX                SUBTRACT 1
812B DF 07          STX     TONECT     PUT IT BACK
812D 26 E8          BNE     TONOUT     SEND ANOTHER
812F CE 0056 UNISPC LDX     #2*DTCT    UNIT SPACE
8132 8D D3   UNSP   BSR     UNITIM     500 MICRO
8134 09             DEX                SUBTRACT 1
8135 26 FB          BNE     UNSP       EQUAL 0?, NO DO AGAIN
8137 39             RTS                YES, RETURN
             *
8138 8D F5   UNISP2 BSR     UNISPC     1 UNIT SPACE
813A 8D F3          BSR     UNISPC     1 UNIT SPACE
813C 39      END    RTS                RETURN TO CALLER
             *
813D 8D F9   WRDSPC BSR     UNISP2     2 SPC UNITS
813F 8D F7          BSR     UNISP2     "  "  "
8141 8D F5          BSR     UNISP2     "  "  "
8143 DE 05          LDX     XTEMP
8145 BD 806F SENCO  JSR     TXON
8148 A6 00          LDA A   0,X        GET A CHAR
814A 08             INX                MOVE POINTER
814B DF 05          STX     XTEMP      SAVE POINTER
814D 81 04          CMP A   #4         IS IT EOT?
814F 27 EB          BEQ     END        YES, GO TO END
8151 81 21          CMP A   #$21       NO. IS IT A SPACE?
8153 27 E8          BEQ     WRDSPC     YES, DELAY WRD
8155 48      SHIFT  ASL A              NO, SHIFT OVER
8156 36             PSH A              SAVE A ON STACK
8157 24 0D          BCC     DITOUT     CARRY CLEAR?
8159 8D B7          BSR     DAH        NO, SEND DAH
815B 32      ENCHA  PUL A              RESTORE A
815C 81 80          CMP A   #$80       END OF CHAR?
815E 26 F5          BNE     SHIFT      NO. SHIFT OVER
8160 8D D6          BSR     UNISP2     2 SPACE UNITS
8162 DE 05          LDX     XTEMP      RESTORE POINTER
8164 20 DF          BRA     SENCO      CONTINUE
             *
8166 8D A5   DITOUT BSR     DIT        SEND DIT
8168 20 F1          BRA     ENCHA      CHECK END CHAR
             *
816A 8D 09   DEL5   BSR     DELONE     DELAY 1 SEC
816C 8D 07          BSR     DELONE
816E 8D 05   DEL3   BSR     DELONE
8170 8D 03          BSR     DELONE
8172 8D 01          BSR     DELONE
8174 39             RTS
             *
```

```
8095 27 0A              BEQ     N2          NC, GET OUT
8097 DE 01              LDX     MINTEN      YES, GET TIME
8099 09                 DEX                 DEC IT
809A DF 01              STX     MINTEN      PUT IT BACK
809C 26 03              BNE     N2          IS IT=0?
809E 7F 0009            CLR     TXFLG       YES, CLR FLG
80A1 DE 03       N2     LDX     IDFIV       I.D. COUNT
80A3 09                 DEX
80A4 DF 03              STX     IDFIV
80A6 26 0B              BNE     OUT1        IS IT =0?
80A8 CE 8190            LDX     #IDENT      IDENTIFY CODE
80AB BD 8145            JSR     SENCC       YES, SEND I. D.
80AE CE 01B0            LDX     #$1B0       5 MIN. WORTH
80B1 DF 03              STX     IDFIV       RESET CLOCK
80B3 B6 4004     OUT1   LDA A   PIAD        CLEAR INT
80B6 F6 4006            LDA A   PIBD        CLEAR INT
80B9 3B                 RTI
                 *
80BA CE B290     TAIL   LDX     #$B290      1 SEC
80BD 8D 10              BSR     TLDLA       GO DELAY (1)
80BF BD 8184            JSR     BEEP1       PAUSE SIGNAL
80C2 CE FFFF            LDX     #$FFFF
80C5 8D 13              BSR     TLDLAY      GO DELAY (2)
80C7 CE 5560            LDX     #$5560
80CA 8D 0E              BSR     TLDLAY
80CC 7E 8022            JMP     RUN         START OVER
                 *
80CF F6 4006     TLDLA  LDA B   PIBD        READ INPUTS
80D2 C4 01              AND B   #1          READ RX BIT
80D4 26 AA              BNE     TLGATF      TAILGATED
80D6 09                 DEX                 DELAY 1 SEC
80D7 26 F6              BNE     TLDLA       CONTINUE
80D9 39                 RTS
                 *
80DA F6 4006     TLDLAY LDA B   PIBD        INPUTS
80DD C4 01              AND B   #1          RECEIVER BIT
80DF 26 9A              BNE     RECON       CONTINUE
80E1 09                 DEX                 DECREMENT
80E2 26 F6              BNE     TLDLAY      MORE DELAY
80E4 39                 RTS
                 *
80E5 F6 4004     TXOFF  LDA B   PIAD        READ CONDX
80E8 C4 7F              AND B   #$7F        MASK
80EA F7 4004            STA B   PIAD        TURN IT OFF
80ED 7F 0009            CLR     TXFLG
80F0 39                 RTS
                 *
80F1 BD 817E     WARN6  JSR     BEEP3       3 BEEPS
80F4 BD 817E     WARN3  JSR     BEEP3       3 BEEPS
80F7 8D 75              BSR     DEL3        DELAY 3 SEC
80F9 8D EA       HOLDOF BSR     TXOFF
80FB F6 4006            LDA B   PIBD
80FE C4 01              AND B   #1          READ RX BIT
8100 26 F7              BNE     HOLDOF      GET RX BIT
8102 8D 66              BSR     DEL5        DELAY 5 SEC
8104 7E 8022            JMP     RUN         RX OFF, GO START
                 *
8107 C6 4A       UNITIM LDA B   #$4A        (500 MICROSEC)
8109 5A          UNITI  DEC B               SUBTRACT 1
810A 26 FD              BNE     UNITI       NOT=0, REPEAT
```

```
8175 CE 07A6     DELONE LDX     #$7A6       1958 CTS
8178 8D 8D       DELX   BSR     UNITIM      500 MICRO
817A 09                 DEX                 SUBTRACT 1
817B 26 FB              BNE     DELX        DO IT AGAIN, SAM!
817D 39                 RTS
                 *
817E CE 8189     BEEP3  LDX     #BEEPR3     POINT CODE
8181 8D C2              BSR     SENCO       SEND IT
8183 39                 RTS
                 *
8184 CE 818D     BEEP1  LDX     #BEEPR1     POINT
8187 20 F8              BRA     *-6         SEND IT
                 *
                 *   DATA
                 *
8189 21          BEEPR3 FCB     $21,$40,$21,$40
818A 40
818B 21
818C 40
818D 21          BEEPR1 FCB     $21,$40,4
818E 40
818F 04
8190 70          IDENT  FCB     $70,$50,$0C,$60,$70,$78
8191 50
8192 0C
8193 60
8194 70
8195 78
8196 21                 FCB     $21,$48,$B0,$48,$21,$28,$48,4
8197 48
8198 B0
8199 48
819A 21
819B 28
819C 48
819D 04
819E 21          RESET  FCB     $21,$68,$50,$D0,$E0,$21,$88,$B8
819F 68
81A0 50
81A1 D0
81A2 E0
81A3 21
81A4 88
81A5 B8
81A6 21                 FCB     $21,$B0,$0C,$90,$28,4
81A7 B0
81A8 0C
81A9 90
81AA 28
81AB 04
                 *
                 *
83F8                    ORG     $83F8
83F8 0002               RMB     2
83FA 0002               RMB     2
83FC 8085               FDB     TIMID
83FE 8000               FDB     START
                 *
                        END
```