RX-320 PC-RADIO Programmer's Reference Guide

Contents

COMMENTS & SUGGESTIONS

Any comments or suggestions regarding this document or the RX-320 firmware should be directed by e-mail to pcradio@tentec.com or by mail to Ten-Tec 1185 Dolly Parton Parkway, Sevierville, TN 37862. The most recent released information will likely be available for download from the Ten-Tec Web page at www.tentec.com.

Convention used in this text

Numeric Types:

0x0A & 0Ah	both refer to a hexadecimal number.
10	refers to a decimal number.
'A'	refers to the ASCII code for the given character. For example, the code for ASCII 'A' is 0x41
(int)	refers to the integer protion of the number that follows
mod	refers to the modulus function which returns the remainder of a division.

A Hardware/Software Radio

The Ten-Tec RX-320 DSP HF receiver is part of a new breed of PC controlled communications equipment that is more *software* than *hardware*. The RX-320 contains state of the art RF and Digital Signal Processing (DSP) circuitry able to receive radio signals and convert them to audio. Unlike conventional tabletop or portable radio receivers the RX-320 contains no front panel controls. The receiver hardware relies entirely on an external controller to provide the user-friendly, radio-like functionality common in today's radios.

Ten-Tec provides a Windows based Graphical User Interface (GUI) which allows the RX-320 to be operated from a PC running Windows 3.1, 95 or 98. As mentioned earlier, the radio-like features of the software are created in the PC rather than the radio. Ten-Tec also encourages and supports the efforts of other software vendors in adding support for the RX-320 to their programs. You may want to check with your favorite software provider to see if RX-320 support is already available. In addition, skilled users may wish to create their own programs to control the RX-320 or have a unique application not supported by commercially available programs. Ultimately it will be necessary for the developer to communicate with the RX-320 attached to the PC. Ten-Tec has compiled the following information, examples and references as a starting point for software developers undertaking the PC-to-RX-320 interface.

The PC to Radio Connection

Interfacing a radio receiver to a PC is not a new concept and manufacturers have developed many different hardware & software approaches over the years. Those experienced with other interfaces may notice that the RX-320 interface is considerably different from the rest. A different approach was required because of system design constraints and the obvious lack of front panel controls. Also, those operations better performed by a PC are moved to the PC to lighten the processing load on the radio. Doing so makes the RX-320 even more dependent on the controlling program. An RX-320 receiver is not really a radio without a properly operating control program.

Hardware Vs Software

The heart of the RX-320 is the DSP processor which executes instructions contained in firmware. The DSP is used for so many tasks that it can be difficult to understand where software ends and hardware begins. However, for developing RX-320 applications an understanding of the relationship between hardware and software is important.

The *hardware* of the RX-320 consists of two printed circuit boards in addition to an external plug-in power supply. The analog receiver functions are supplied by circuitry located on the RF board. The RF board contains fixed and tunable mixing stages and provides the initial receiver selectivity by passing signals through intermediate narrow band stages. After processing by the RF board the signals are passed to the DSP board where they are further filtered, detected, measured, amplified and controlled.

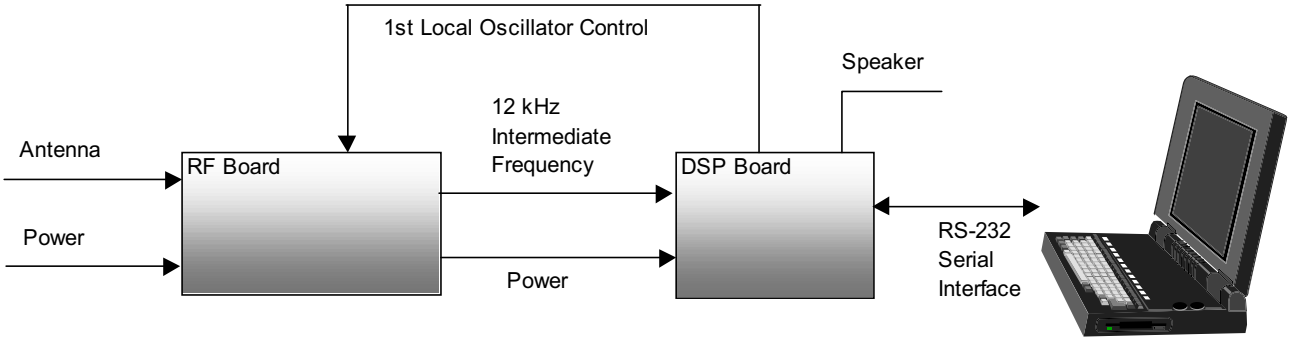

FIGURE 1 - RX-320 DSP/RF/PC Relationship

The DSP board also has the responsibility of managing the communications between the radio and the PC. The 9-pin serial port on the RX-320 provides bi-directional RS-232 level communications between the radio and PC. The communication parameters are fixed at 1200 baud, 8 data bits and no parity. The interface is operated at low speed due to design limitations but using a compact command set described later minimizes the impact on performance. It is quite common for RS-232 serial communications to be handled in hardware by a UART (Universal Asynchronous Receiver Transmitter). However, in the RX-320 the DSP performs those operations in a software version of a UART.

For some functions the RF and DSP boards have to work together. For example, tuning the receiver to a particular frequency is managed through the proper setting of both the RF boards PLL (Phase Locked Loop) and the DSP processor's internal registers. The DSP board has complete control over the RF board's PLL settings but the program running on the PC is required to pre-calculate the factors necessary for tuning. Methods for calculating the particular factors are described later.

Overview of the RX-320 Command Set

In a conventional receiver the operator controls the radio by way of front panel controls. The user only has to turn a knob or push a button. Usually a microprocessor hidden inside the radio will react to the user's motions and then manipulate the radio hardware to get a pre-defined result. The RX-320 lacks a front panel and the DSP processor has very little pre-defined functionality. It is up to the controlling program to provide the operator's knobs & buttons and manipulate the DSP to create the functionality. In this arrangement the controlling program is responsible for pre-processing the user's request and determining the necessary DSP settings to achieve the end result. Many DSP functions are inter-related and simply tuning the receiver may require adjustment of all DSP settings.

The instruction set for programming the RX-320 is limited to 7 commands. As previously indicated many parameters are inter-related. In general an RX-320 instruction is a single command letter which may be followed by data and then terminated by a carriage return <cr>. Unless otherwise noted the data is in hex format. Using a hex format requires fewer bytes when compared to an ASCII format. Some radio interfaces use a BCD (Binary Coded Decimal) format. BCD format is quite useful for equipment with a front panel because many display devices will require BCD. The RX-320 operates with hex data and all commands that require numeric data will need that data to be in hex format. In hex format a number like 25 decimal is represented as 19 hexadecimal or 19h or sometimes written as 0x19. All refer to the same number.

The DSP provides control over **MODE, FREQUENCY, BFO, FILTER, AGC MODE, SPEAKER LEVEL and LINE OUTPUT LEVEL**. In addition, the DSP can respond to requests for **SIGNAL STRENGTH** (limited) and **DSP FIRMWARE** revision number. The RX-320 contains no provision for storing any parameters so the radio must be reprogrammed whenever the radio is turned on and/or when the controlling program is started. Reprogramming the radio at power-up requires setting the MODE, FERQUENCY, FILTER and VOLUME level. To prevent unwanted audio output the VOLUME setting should be done last.

In general, the RX-320 will accept any provided data as being valid. It is up the programmer to ensure that the supplied data is correct. Where a command has a limited number of data option, such as the AGC MODE command, failure to provide a valid selection will result in the radio choosing the default selection. There will be no notification that the data is invalid. Where a command is totally unrecognized by the RX-320 the radio will send back a response consisting of a single letter 'Z' followed by a carriage return <cr>.

Power-up Notification

When power is applied or when the power switch is set to ON the radio will transmit the string 'DSP START <cr>' over its serial interface. A controlling program can watch for this transmission and will know that the RX-320 will need to be reprogrammed. At power-up the receiver audio will be muted and will need to have all of its parameters programmed before it will operate properly. This involves setting the tuning factors, selecting a filter and setting the audio levels.

MODE

The radio supports AM, LSB, USB and CW detection modes. The selected MODE must be considered in calculations for the FREQUENCY tuning factors so any change in detection mode will require a recalculation of the frequency tuning factors.

format: 'M' mc <cr>
where: 'M' = the ASCII M (0x4d) character 0x4D
mc = possible detection mode
ASCII '0' (0x30) for AM mode
ASCII '1' (0x31) for USB mode
ASCII '2' (0x32) for LSB mode
ASCII '3' (0x33) for CW mode
<cr> = ASCII carriage return 0x0D
response: none.

example: M0 0x0D for AM detection mode
M1 0x0D for USB detection mode

FILTER

The RX-320 contains a large number of selectable filters that can be used in any detection mode. There is no default filter selected and the appropriate filter should be set for the selected detection mode. The selected filter must be considered in calculations for the FREQUENCY tuning factors so any change in filter selection will require a recalculation of the frequency tuning factors. In general the filter selection will also affect the BFO setting factor. The exception is AM mode, where the BFO setting has no meaning and is ignored.

format: 'W' fn <cr>
where: 'W' = the ASCII W (0x57) character
fn = a filter number in the range 0 thru 33 (binary). From Table 1 below.
<cr> = ASCII carriage return 0x0D
response: none.
example: W 0x10 0x0D for filter number 16

Filter #	Bandwidth
0	6000
1	5700
2	5400
3	5100
4	4800
5	4500
6	4200
7	3900
8	3600
9	3300
10	3000
11	2850

Filter #	Bandwidth
12	2700
13	2550
14	2400
15	2250
16	2100
17	1950
18	1800
19	1650
20	1500
21	1350
22	1200
23	1050

Filter #	Bandwidth
24	900
25	750
26	675
27	600
28	525
29	450
30	375
31	330
32	300
33	8000

TABLE 1 - Available Filter Selections

SPEAKER & LINE OUTPUT VOLUME

The RX-320 has two audio outputs. Each can be controlled seperately or may be controlled together. The speaker volume control sets the level of an optional speaker which may be connected to speaker jack located on the back of the RX-320. The Line level settings control the amplitude of the low level line output located on the rear panel of the RX-320. The line-level output can be connected to the line-in of a PC sound card or to an amplified speaker such as those that can be connected to a portable CD player.

The commands refer to a volume level but the represent an attenuation code that is passed to the DSP processor. The range is 0 thru 63 where 0 represents the loudest setting and 63 represents the lowest setting. each step is equal to 1.5 dB of attenuation for a total control range of 96 dB.

format: CC xxb vab <cr>
where: CC = Control character
ASCII 'V' (0x56) for speaker volume control
ASCII 'A' (0x41) for line-out control
ASCII 'C' (0x43) for both
xxb = don't care byte, used as a place holder
vab = volume attenuation number 0-63 (binary)
<cr>=ASCII carriage return (0x0D)

response: none
example: V 0x00 0x20 0x0D sets the speaker to level 32

AGC MODE

The RX-320 can operate in SLOW, MEDIUM or FAST AGC mode. At power-up the radio will default to MEDIUM AGC mode which will provide the most versatile response. The AGC mode may be changed at any time to take advantage of changing band conditions.

format: 'G' CC <cr>
where: 'G' = ASCII G (0x47) character
CC = Control Character
ASCII '1' (0x31) for SLOW
ASCII '2' (0x32) for MEDIUM (default)
ASCII '3' (0x33) for FAST
response: none.
example: G1 0x0D for SLOW AGC Mode.

FREQUENCY

The RX-320 does not have the ability to accept a frequency command directly. The radio can only accept tuning factors, which are the result of calculations made by the PC. There are 3 factors involved, the coarse-tuning factor, the fine-tuning factor and the BFO factor. The need for the different factors arises from the fact that the RF board provides a portion of the tuning while the remainder is provided by the fine-tuning and BFO mixers in the DSP processor.

Figure 2 shows a diagram of the RX-320's mixing scheme. The dotted vertical line shows the break between Analog (RF board) and DSP processes. Table 2 gives some example frequencies using the mixing scheme of figure 2. As indicated on the diagram, the first local oscillator, LO1, is tunable in 2.5 kHz steps. In General, ignoring correction factors, the coarse tuning factor results from the need to tune LO1 as close as possible to the receive desired frequency. The fine-tuning factor represents the difference between the desired frequency and the frequency resulting from where LO1 will be tuned.

For example, given a desired frequency of 12.001 MHz and a 2.5 kHz tuning step we could tune to 12.000 MHz. The remaining 0.001 MHz would be passed to the fine-tuning mixer and LO3. The equations shown on page 8 show dividing the frequency given in MHz by 0.0025 (2500 Hz). The integer portion of the result is the basis of the coarse tuning factor while the fractional portion is the basis of the fine-tuning factor. Given the example above the 12.001/0.0025 = 4800.4. The 4800 applies to the coarse tuning factor and 0.4 applies to the fine-tuning factor. Again, LO1 provides the coarse tuning in 2.5 kHz chunks and the DSP provides the fine-tuning.

LO4, also referred to as the BFO, supplies an additional mixer that is used to provide additional post-filtering frequency translation in some detection modes.

The discussion and examples presented thus far have ignored a variety of correction factors that are needed to make the whole thing work. For example, LO1 tunes from 45 – 75 MHz, which is 45 MHz away from the desired frequency. This fact is taken into account by the 18000 (45.00/0.0025) which is added to the integer result of the basic calculation. Other adjustments arrise from design constraints while still other arrise from the selection of detection mode or filter bandwidth.

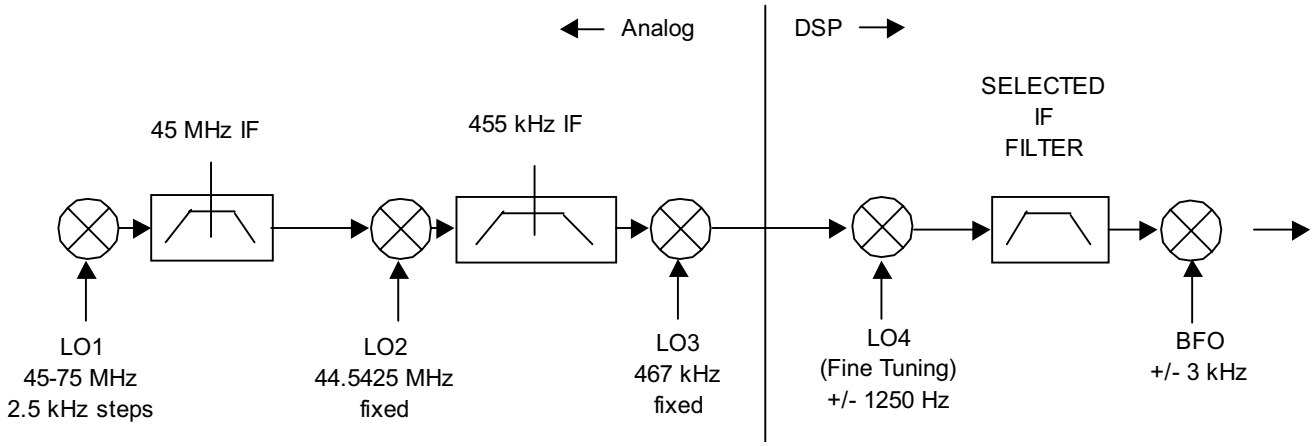

Figure 2 - RX-320 Mixing Scheme

Freq	LO1	NVAL	IF1	LO2	IF2	LO3	IF3	LO4
0.100000	45.09750	18039	44.9975	44.5425	0.455	0.467	0.012	0
0.100100	45.09750	18039	44.9975	44.5425	0.455	0.467	0.012	100
2.000000	46.99750	18799	44.9975	44.5425	0.455	0.467	0.012	0
2.005000	47.00250	18801	44.9975	44.5425	0.455	0.467	0.012	0
5.000000	49.99750	19999	44.9975	44.5425	0.455	0.467	0.012	0
10.001500	54.99750	21999	44.9975	44.5425	0.455	0.467	0.012	1500
11.000010	55.99750	22399	44.9975	44.5425	0.455	0.467	0.012	10
15.000000	59.99750	23999	44.9975	44.5425	0.455	0.467	0.012	0
30.000000	74.99750	29999	44.9975	44.5425	0.455	0.467	0.012	0

Table 2 - Typical Oscillator Frequencies

SETTING THE RADIO TUNING FACTORS

Because the FILTER, MODE and BFO selections all affect the tuning factors they should be adjusted before calculating the new tuning factors. The command format for programming the RX-320's tuning factors is shown below. The controlling program must pre-calculate and format the tuning factors.

format: 'N' Ch Cl Fh Fl Bh Bl <cr>
where: 'N'= is the ASCII 'N' (0x4E) character
Ch = high byte the 16 bit coarse tuning factor
CL = low byte of the 16 bit coarse tuning factor
Fh = high byte of the 16 bit fine tuning factor
Fl = low byte of the 16 bit fine tuning factor
Bh = high byte of the 16 bit BFO factor
Bl = low byte of the 16 bit BFP factor
<cr> = ASCII carriage return (0x0D)
response: none.

The tuning factors Ch, Cl, Fh, Fl, Bh and Bl are calculated using the formulas presented here.
input:

Tfreq = Tuned Frequency in MHz.
Mcor = Mode Correction = 0 for AM mode, +1 for USB, -1 for LSB, -1 for CW.
Fcor = Filter Correction calculated using (Bandwidth/2)+200 where bandwidth is in Hz.
Cbfo = Desired center frequency of filter in Hz for CW mode. Only needed in CW mode.

calculation:

AdjTfreq = Adjusted Tuned Frequency = Tfreq-0.00125+(Mcor*(Fcor+Cbfo))/1000000

Ctf = Coarse tuning factor = (int) (AdjTfreq)/0.0025)+18000
where (int) is the integer function to get the integer only portion of the division

Ftf = Fine tuning factor = mod(AdjTfreq,0.0025) * 2500*5.46
where mod is the modulus function used to get the fractional remainder of a division operation.

Btf = Bfo Tuning Factor = (int) ((fcor+CWBFO+8000)*2.73)
where (int) is the integer function to get the integer only protion of the division

Example Program

Listing 1 contains a BASIC program that demonstrates the calculation and formatting of radio command and associated related data fields. The program will control all functions of an RX-320 receiver but the user interface is limited. This program can be used as a ready-made starting point for developing more sophisticated applications.

Almost any programming language can be used to develope RX-320 applications. This example uses BASIC because of the simplicity of the language. There are several varieties of BASIC available and they should work fine with the exception of the references to the COM port (see line 140 in Listing 1). Some versions of BASIC will support only COM 1 while other support COM 1 and COM 2. Referencing a COM port that is not available will likely result in an error.

Commands Which Request Information

There are two RX-320 commands which will cause the radio to generate a response. The Signal Strength and Firmware Revision can be requested. The command and response formats are described below.

SIGNAL STRENGTH

The RX-320 receiver has the ability to report the relative signal strength of signals it is receiving. When requested, the radio will return a 16 bit unsigned number which represents the relative signal level at the radio's tuned frequency. This level will vary from near 0 to around 10,000 representing a range of approximately 80 dB.

Format: 'X'<cr>
Where: 'X' = ASCII X (0x58)
<cr> = ASCII carriage return (0x0D)

returns: 'X' Hb Lb <cr>
Where: 'X' = ASCII X (0x58) character.
Hb = High Byte of signal strength.
Lb = Low Byte of signal strength.
<cr> = Carriage return (0x0D).

DSP FIRMWARE REVISION NUMBER

The revision number of the firmware contained in the RX-320 can be requested using this command. The radio will respond with a text string that contains the revision number in an integer form. The integer number divided by 100 will give the major and minor firmware revisions. For example: id the radio responds with 106, dividing by 100 yeilds 1.06 which means major firmware revision 1 with minor revision .06 or 1.06.

format: '?' <cr>
where: '?' = ASCII question mark (0x3F) character
<cr> - carraige return (0x0D)

returns: "VER 106" for revision 1.06. Revision number will vary.

Listing 1 - BASIC DEMO PROGRAM

```
10 REM Ten-Tec RX-320 test program
20 LNVOL=25              ' Line-out volume
30 '
40 AMMODE=ASC("0"):CWMODE=ASC("3"):USBMODE=ASC("1"):LSBMODE=ASC("2") ' modes defined
50 DIM FILTERS(34)       ' array to hold filter list
60 GOSUB 1360            ' preload filters[] array
70 D$=CHR$(13)                   ' carriage return defined
80 SPKVOL=30             ' speaker volume
90 MODE=AMMODE                   ' detection mode
100 RADIOFREQ=.93        ' tuned frequency
110 FILTER=0     ' filter number from filter list
120 CWBFO=0
130 '
140 OPEN "com1:1200,n,8,1,rs,ds" AS #2
150 ' main routine
160 CLS
170 GOSUB 320
180 GOSUB 1030
190 GOSUB 440
200 GOSUB 1130
210 C$=INKEY$ : IF C$="" GOTO 210
220 IF C$="1" THEN GOSUB 570:GOTO 150:ELSE
230 IF C$="2" THEN GOSUB 770:GOTO 150:ELSE
240 IF C$="3" THEN GOSUB 590:GOTO 150:ELSE
250 IF C$="4" THEN GOSUB 890:GOTO 150:ELSE
260 IF C$="5" THEN GOSUB 940:GOTO 150:ELSE
270 IF C$="6" THEN GOSUB 990:GOTO 150:ELSE
280 IF C$="7" THEN GOSUB 1260:GOTO 150:ELSE
290 IF C$="8" THEN GOTO 310
300 GOTO 150
310 CLOSE:END
320 ' Tuning Factor Calculations
330 IF MODE=AMMODE THEN MCOR=0
340 IF MODE=CWMODE THEN MCOR=-1
350 IF MODE=USBMODE THEN MCOR = 1
360 IF MODE=LSBMODE THEN MCOR= -1
370 FCOR = FILTERS[FILTER]/2+200
380          ADJFREQ=RADIOFREQ-.00125+(MCOR*((FCOR+CWBFO)/1000000!))
390                  CTVAL=INT(ADJFREQ*400!)
400                  FTVAL=INT(((ADJFREQ*400!)-CTVAL)*2500!*5.46)
410                  CTVAL=CTVAL+18000
420          BTVAL=INT((FCOR+CWBFO+8000)*2.73)
430 RETURN
440 ' Menu
450 'CLS
460 LOCATE 1,15: PRINT "Ten-Tec RX-320 PC RADIO Interface Demonstration"
470 LOCATE 4,20: PRINT "       1 .. Frequency "
480 LOCATE 5,20: PRINT "       2 .. Filter"
490 LOCATE 6,20: PRINT "       3 .. Mode"
500 LOCATE 7,20: PRINT "       4 .. Speaker Volume"
510 LOCATE 8,20: PRINT "       5 .. Line Out Volume"
520 LOCATE 9,20: PRINT "       6 .. Bfo"
530 LOCATE 10,20: PRINT"       7 .. Query DSP Revision #"
540 LOCATE 11,20: PRINT"       8 .. Quit The Program"
550 LOCATE 16,25: PRINT"      Make Selection "
560 RETURN
```

```
570 LOCATE 16,25: INPUT "Enter Frequency (MHz) ";RADIOFREQ
580 RETURN
590 CLS:
600 GOSUB 730
610 LOCATE  4,25: PRINT "Currently in ";MD$
620 LOCATE  6,25: PRINT "1 .. AM Mode"
630 LOCATE  8,25: PRINT "2 .. USB Mode"
640 LOCATE 10,25: PRINT "3 .. LSB Mode
650 LOCATE 12,25: PRINT "4 .. CW Mode"
660 LOCATE 16,25: PRINT "Select New Mode"
670 C$=INKEY$:IF C$="" GOTO 670
680 IF C$="1" THEN MODE=AMMODE
690 IF C$="2" THEN MODE=USBMODE
700 IF C$="3" THEN MODE=LSBMODE
710 IF C$="4" THEN MODE=CWMODE
720 RETURN
730 ' convert MODE code to text name of mode
740 IF MODE=AMMODE THEN MD$="AM MODE" : IF MODE=LSBMODE THEN MD$="LSB MODE"
750 IF MODE=USBMODE THEN MD$="USB MODE" : IF MODE=CWMODE THEN MD$="CW MODE"
760 RETURN
770 ' FILTER SELECTION SUBROUTINE
780 CLS:
790 FOR N=0 TO 10
800 LOCATE N+1,20: PRINT N;" ";FILTERS[N]
810 LOCATE N+1,40: PRINT N+11;" ";FILTERS[N+11]
820 LOCATE N+1,60: PRINT N+22;" ";FILTERS[N+22]
830 NEXT N
840 LOCATE 12,60: PRINT 33;" ";FILTERS[33]
850 LOCATE 16,25 : INPUT "Enter Fileter Number (0-33) ";FILTER
860 IF FILTER <0 GOTO 850
870 IF FILTER >33 GOTO 850
880 RETURN
890 ' SPEAKER VOLUME
900 LOCATE 16,25:INPUT "Speaker Volume Setting (0-63) (soft->loud) ";SPKVOL
910 IF SPKVOL<0 GOTO 900
920 IF SPKVOL>63 GOTO 900
930 RETURN
940 ' LINE OUT VOLUME
950 LOCATE 16,25: INPUT "Line Out Volume Setting (0-63) (soft->loud) ";LNVOL
960 IF LNVOL<0 GOTO 950
970 IF LNVOL>63 GOTO 950
980 RETURN
990 LOCATE 16,25: INPUT "Enter Bfo Frequency (0-2000Hz) ";CWBFO
1000 IF CWBFO<0 GOTO 990
1010 IF CWBFO>2000 GOTO 990
1020 RETURN
1030 'SUBROUTINE TO FORMAT CALCULATED PARAMETERS AND SEND TO RADIO
1040 NH=INT(CTVAL/256):NL=CTVAL-NH*256
1050 FH=INT(FTVAL/256):FL=FTVAL-FH*256
1060 BH=INT(BTVAL/256):BL=BTVAL-BH*256
1070 C$="W"+CHR$(FILTER)+D$:PRINT #2,C$;
1080 C$="N"+CHR$(NH)+CHR$(NL)+CHR$(FH)+CHR$(FL)+CHR$(BH)+CHR$(BL)+D$:PRINT #2,C$;
1090 C$="A"+CHR$(127)+CHR$(63-LNVOL)+D$:PRINT #2,C$;
1100 C$="V"+CHR$(127)+CHR$(63-SPKVOL)+D$:PRINT #2,C$;
1110 C$="M"+CHR$(MODE)+D$:PRINT #2,C$;ELSE
1120 RETURN
```

```
1130 'display current settings
1140 GOSUB 730
1150 LOCATE 20,2 : PRINT "Freq= ";RADIOFREQ
1160 LOCATE 21,2 : PRINT "Mode= ";MD$
1170 LOCATE 22,2 : PRINT "Filt= ";FILTERS(FILTER)
1180 LOCATE 20,25: PRINT "spkr= ";SPKVOL
1190 LOCATE 21,25: PRINT "Line= ";LNVOL
1200 LOCATE 22,25: PRINT " Bfo= ";CWBFO
1210 LOCATE 20,45: PRINT "Coarse Tune Factor= ";CTVAL
1220 LOCATE 21,45: PRINT "  Fine Tune Factor= ";FTVAL
1230 IF MODE=AMMODE THEN BTVAL=0          ' btval is ignored by RX-320 in AM MODE
1240 LOCATE 22,45: PRINT "   Bfo Tune Factor= ";BTVAL
1250 RETURN
1260 ' Get DSP VERSION #
1270 IF LOC(2)<>0 THEN INPUT#2,C$                    ' read port first to empty any old data
1280 C$="?"+D$:PRINT#2,C$;
1290 GOSUB 1430
1300 IF LOC(2)=<7 THEN GOTO 1300
1310 VER$=INPUT$(LOC(2),2)
1320 LOCATE 16,26: PRINT "DSP REPORTS ";VER$
1330 LOCATE 18,26 : PRINT "Press Any Key To Continue"
1340 C$=INKEY$:IF C$="" GOTO 1330
1350 RETURN
1360 ' load filter array
1370 FOR N=0 TO 33 STEP 1
1380 READ FILTERS(N)
1390 NEXT N
1400 RETURN
1410  DATA  6000,5700,5400,5100,4800,4500,4200,3900,3600,3300,3000,2850,2700,2550,2400,2250,2100,1950,1800
1420 DATA 1650,1500,1350,1200,1050, 900, 750, 675, 600, 525, 450, 375, 330, 300,8000
1430 ' DELAY ROUTINE
1440 FOR D=1 TO 20000
1450 NEXT D
1460 RETURN
```

NOTES