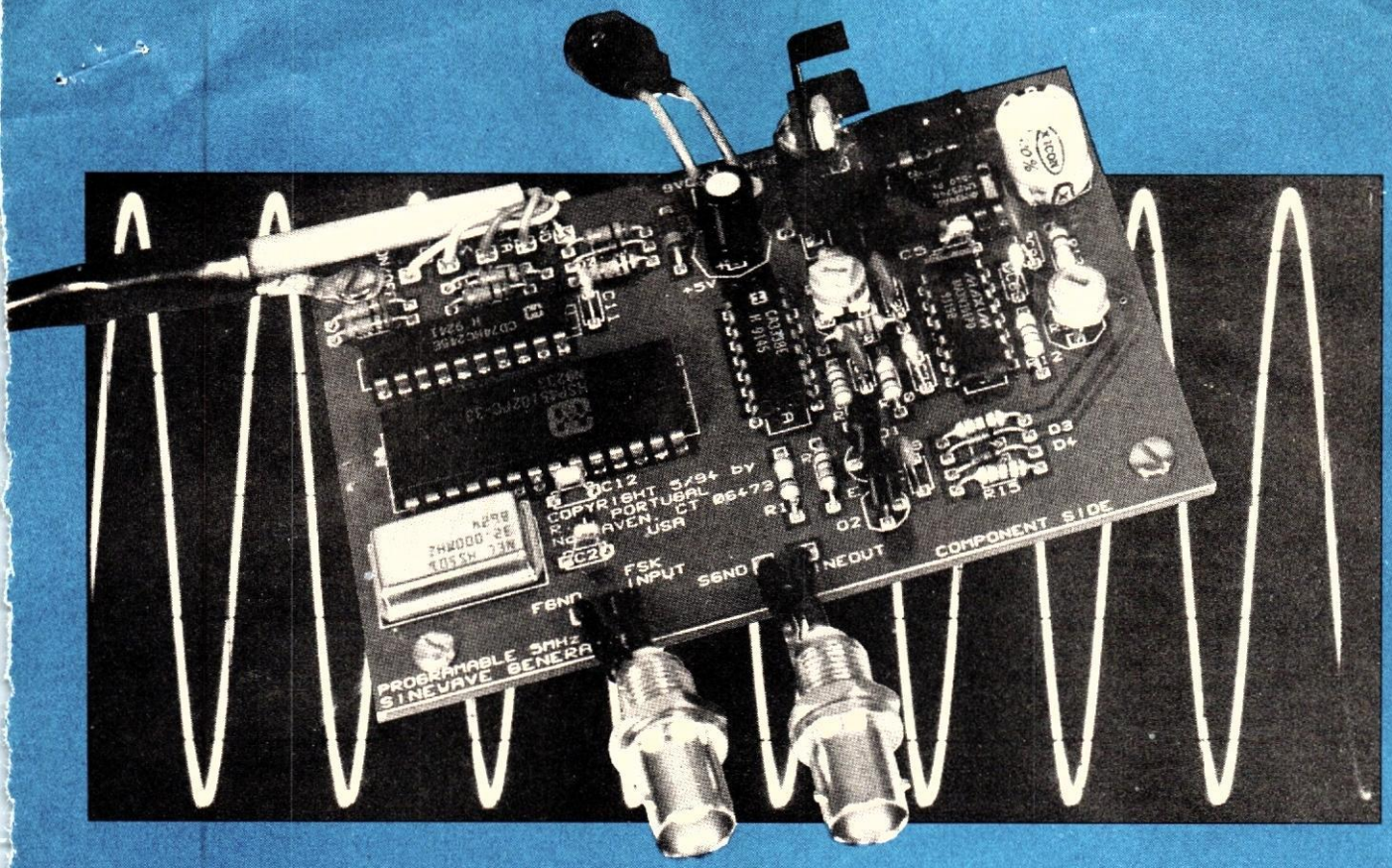


PROGRAMMABLE SINEWAVE GENERATOR

RONALD J. PORTUGAL

Build this programmable sinewave generator once, and you can program it forever!

A PROGRAMMABLE SINEWAVE GENERATOR can be a lot more accurate than an analog-based generator. When teamed with a PC or single-board computer, a programmable generator becomes an application-specific instrument offering a limitless variety of software-based "front panels" without the need to modify the hardware. It's a fun device to build and experiment with. Plus, you have to build it only once, but you can program it forever.

There are many tasks that can be performed with a programmable sinewave generator (PSG):

- Convert RS-232 signals to audio or RF frequency FSK (frequency shift keying) format for phone line or wireless modem transmitters
- Function as a signal source for bode plots
- Function as a linear, log, discrete, or programmed sweep generator
- Act as a local oscillator and carrier frequency generator for a transceiver
- Function as a general-purpose generator for industry, education, and experimenter labs
- Synthesize single-voice music compositions
- Generate telephone keypad tones
- Synthesize voice or sound
- Act as a general-purpose secondary frequency standard
- Generate frequency bursts
- Generate continuous-wave amplitude modulation
- Provide a stable, accurate voltage-controlled oscillator for amateur radios
- Generate automatic-test-equipment sweep signals

A computer's parallel port is ideal for controlling a PSG because the parallel port is easy to use, and there is one in most personal computers made today—including most laptop models. Although the serial port could be used, the required

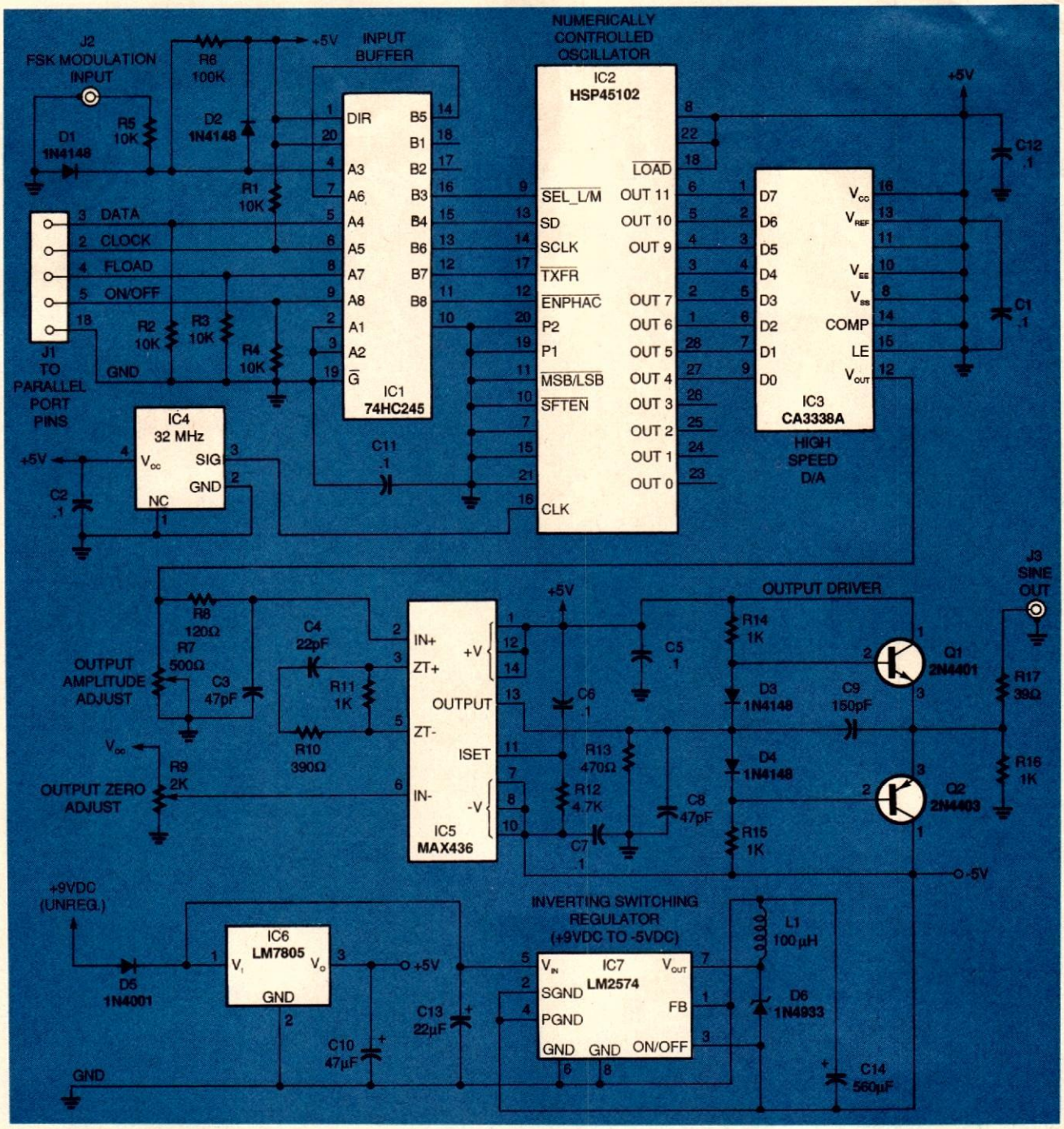


FIG. 1—PSG SCHEMATIC. The NCO chip is a serial input device that needs either 32 or 64 bits of data to generate sinewaves; 32 bits are required for single-frequency operation and 64 bits are required for dual-frequency (FSK) operation.

interface would be more complex, hardware-intensive, and expensive.

The heart of the PSG is an HSP45102 numerically controlled oscillator (NCO). The frequency to be generated is selected from two frequency control words. A single control pin selects the word that determines the output frequency. Switching from one frequency to another occurs in one clock cycle, with a six-clock-pulse pipeline delay from the time that the new control word is loaded until the new frequency appears at the NCO's output. Twelve-bit binary words selected from the chip's internal memory are fed to a Harris CA3338A digital-to-analog converter (DAC) that generates the synthesized sinewave output.

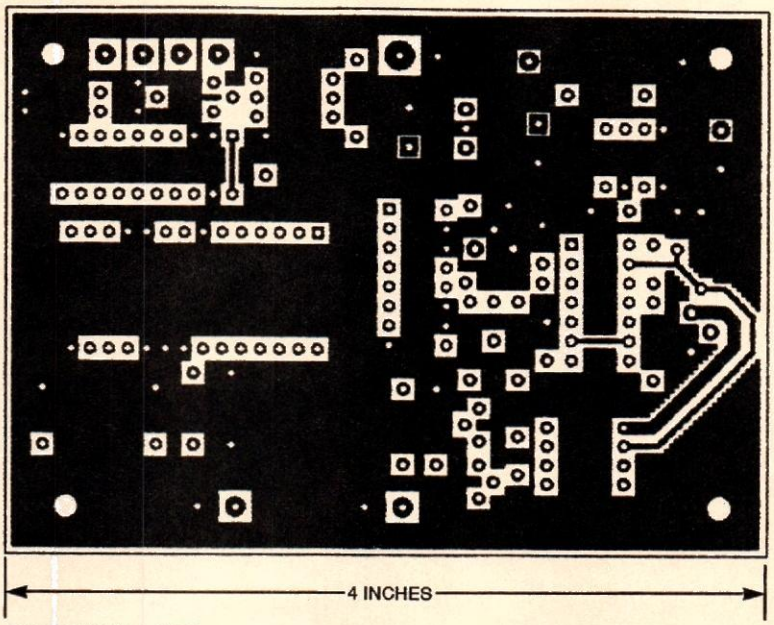
Once the sinewave has been synthesized, it is level-shifted to be symmetrical around zero volts and then filtered to remove the remaining components of the high-frequency NCO clock frequency (32 MHz). Finally, the sinewave is buffered and fed to the PSG's output.

How does it work?

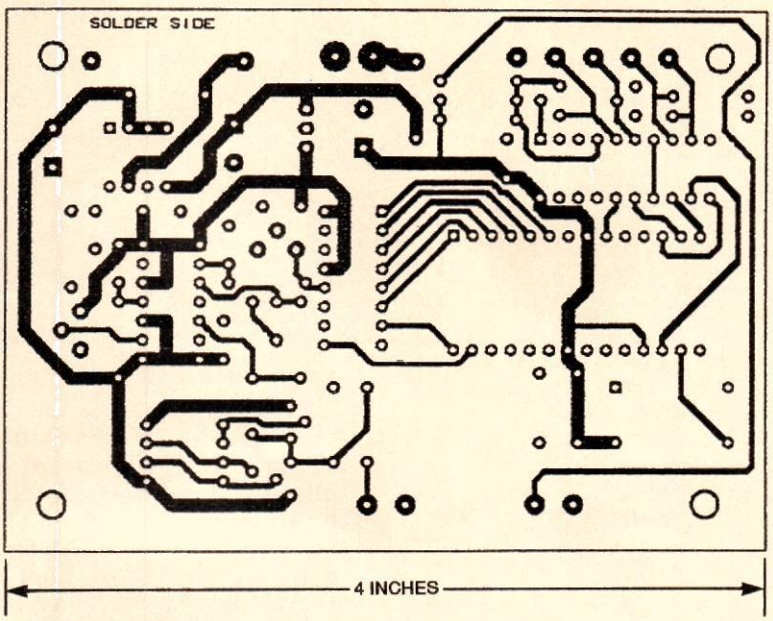
Refer to the upper left-hand corner of the schematic, Fig. 1. The DATA, CLOCK, FLOAD, and ON/

TABLE 1

PARALLEL PORT DB25 PIN NO.	9	8	7	67	5	4	3	2
DATA PORT BIT	D7	D6	D5	D4	D3	D2	D1	D0
BIT WEIGHT	128	64	32	16	8	4	2	1
BIT FUNCTION	X	X	X	X	ON/OFF	FLOAD	DATA	CLOCK



PSG COMPONENT SIDE.



PSG SOLDER SIDE.

OFF input lines couple the board to a PC's parallel port. Each of the four inputs is buffered by IC1 and then fed to the HSP45102 NCO chip. The NCO chip is a serial input device that needs either 32 or 64 bits of data to start generating sinewaves; 32 bits are required for single-frequency operation and 64 bits are required for dual frequency shift keying (FSK).

A shift-clock pulse is generated by the PC with each data bit fed to the DATA input, and it is fed to the CLOCK input of the NCO. When all 32 (or 64) data bits have been fed to the NCO, the FLOAD line is activated and the new frequency(s) become available at the output of the generator. The ON/OFF control line turns the output signal on or off under program control.

The TTL-compatible FSK modulation input selects one of the two programmed frequencies to be seen at the output of the instrument. For example, assume the two programmed frequencies are 1.0001 megahertz and 0.9999 megahertz . A high level (TTL logic 1) at the FSK input would cause the 1.001-megahertz signal to appear at the output, while a low level (TTL logic 0) would cause the 0.999 megahertz signal to show up at the output. The switching delay between signals is six clock pulses—in this case, 187.5 nanoseconds.

The FSK input is held high by 100K resistor R6 to ensure that the first 32 bits of input data activate the NCO. The FSK input is protected from negative voltages by D1 and voltages in excess of +5 volts by D2.

When the NCO has received valid data, it computes the location of the sinewave amplitudes (stored in ROM) needed to generate the output signal. For example, if a 1-megahertz sinewave has been requested, the NCO computes the location of the 32 amplitude values necessary to make up the 1-megahertz sinewave (F_{CLK}/F_{OUT} = number of amplitude values) and then sends the binary values in a 12-bit format to 12 output lines at a 32-megahertz rate (every 31.25 nanoseconds). A digital-to-analog converter (DAC) connected to the NCO's output lines converts the amplitude values to analog voltages. A 12-bit, 31.25-nanosecond-settling-time DAC is needlessly expensive, so an 8-bit DAC is substituted. The DAC's vertical

resolution, better than 0.4%, is adequate for this application.

The amplitude of the DAC output signal is controlled by R7 while the sinewave DC zero level is set by R9. The DAC output (0 to +5 volts unloaded) must be level-shifted to produce a sinewave symmetrical around zero volts. It also must be filtered to produce clean sinewaves.

Because the number of samples per sinewave cycle is equal to the ratio of the clock frequency and the desired output frequency, it can be seen that at 5 megahertz, the number of samples forming the sinewave will be 6.4 (32 MHz/5 MHz), which is a pretty "steppy" sinewave.

The two problems, filtering and level-shifting, are solved by the Maxim MAX436 wideband transconductance amplifier IC5. A transconductance amplifier is a true differential amplifier with high-impedance inputs. It provides accurate gain without negative feedback, eliminating closed-loop phase shift. This is the main cause of oscillation in voltage-output, high-speed amplifiers. The output of the transconductance amplifier is a current that is proportional to the differential input voltage, and the amplifier is virtually short-circuit proof. The gain of the circuit is set by the ratio of two external impedances and an internal current gain factor K:

$$\text{Gain} = K(Z_{LOAD}/Z_T)$$

The device also has a bandwidth in the 275-megahertz range with an 800-volt-per-microsecond slew rate.

For level-shifting, R9 is connected between +5 volts and ground with its center tap fed to the negative input terminal of the MAX436. Simply adjust R9 so that the output signal is symmetrical about ground.

Three resistive-capacitive (RC) sections are distributed throughout the circuit. The first section consists of the output impedance of DAC IC3, R7, R8, and C3. The second is comprised of R13, the output impedance of the transconductance amplifier, the input impedance of the complementary

PARTS LIST

All resistors are ¼-watt, 5%, unless noted.

R1–R5—10,000 ohms
R6—100,000 ohms
R7—500 ohms, ¼-inch round, single-turn cermet trimmer potentiometer
R8—120 ohms
R9—2000 ohms, ¼-inch round, single-turn cermet trimmer potentiometer
R10—390 ohms
R11, R14–R16—1000 ohms
R12—4700 ohms
R13—470 ohms
R17—39 ohms
C1, C2, C5–C7, C11, C12—0.1 µF, ceramic
C3, C8—47 pF, 5%, ceramic NPO
C4—22 pF, 5%, ceramic NPO
C9—150 pF, 5%, ceramic NPO
C10—47 µF, 25 volts, electrolytic
C13—22 µF, 25 volts, electrolytic
C14—560 µF, 16 volts, electrolytic

Semiconductors

IC1—74HC245 octal non-inverting bus transceiver
IC2—HSP45102 12-bit numerically controlled oscillator (Harris)
IC3—CA3338A 8-bit DAC (Harris)
IC4—NEL HS501 32-MHz clock oscillator module (Mouser No. 332-3320 or equivalent)
IC5—MAX436 wide-band transconductance amplifier (Maxim)
IC6—7805, 5-volt regulator (TO-220 package, Motorola or equivalent)
IC7—LM2574-5 inverting switching

LISTING 1

```
'*************************************************************************
'                          PSG CONTROL PROGRAM                            *
'       COPYRIGHT 5/94 by R.J. PORTUGAL, NORTH HAVEN, CT. 06473, U.S.A.   *
'*************************************************************************
DECLARE SUB FREQUENCY (L1, L2, FLAG)        '**** Frequency input routine
DECLARE SUB FREQCLR (L1, L2)                '**** Clear frequency entries
DECLARE SUB LOAD ()                         '**** Parallel-serial data convert
DECLARE SUB ERMSG (L1, L2)                  '**** Error messages
DECLARE SUB DSPLY ()                        '**** Display screen

COMMON SHARED FMT$, N1&, N2&, K, DP, PORT$, MSG$(), L1, L2
DIM Z AS STRING * 1: DIM MSG$(20)
FMT$ = "###,###,###.###.####": Clk = 32000283: K = (2 ^ 32 / Clk)

'*************************************************************************
'       To select NCO parallel printer remove the (') "comment symbol"
'       in front of the appropriate program line.
'       PRINTER PORT "LPT2" will be selected by the following:

     'port$ = "LPT1": dp = &H3BC     '*** LPT1 DATA PORT ADDRESS
      PORT$ = "LPT2": DP = &H378     '*** LPT2 DATA PORT ADDRESS
     'port$ = "LPT3": dp = &H278     '*** LPT3 DATA PORT ADDRESS
'*************************************************************************
'********** (Message list) **********
MSG$(1) = "          Input range is from 0.00Hz to 10,000,000.00Hz          "
MSG$(2) = "     Program accepts numerals and a single decimal point         "
MSG$(3) = STRING$(64, " "): MSG$(4) = STRING$(21, " ")
MSG$(5) = "Last input not transferred to NCO": MSG$(6) = STRING$(50, " ")
MSG$(7) = "PSG output signal OFF"
MSG$(8) = "PROGRAMMABLE SINEWAVE GENERATOR CONTROL SCREEN"
MSG$(9) = "PSG printer port is " + PORT$ + ".  "
MSG$(10) = PORT$ + " uses I/O port " + HEX$(DP) + "HEX"
MSG$(11) = "COPYRIGHT 5/94 by R.J. PORTUGAL, NORTH HAVEN, CT. 06473, U.S.A."

COLOR 15, 1: CLS : CALL DSPLY: N1& = 0: N2& = 0: CALL LOAD
'-------------------------------------------------------------------------
DO: K$ = INKEY$                                 '*** Main Program
IF LEN(K$) = 2 THEN         'check keyboard entry for 2 character scan code
     IF ASC(RIGHT$(K$, 1)) = 59 THEN CALL FREQUENCY(5, 6, 1)    '**** [F1]
     IF ASC(RIGHT$(K$, 1)) = 60 THEN CALL FREQUENCY(8, 9, 2)    '**** [F2]
     IF ASC(RIGHT$(K$, 1)) = 61 THEN                            '**** [F3]
       N1& = 0: N2& = 0: CALL LOAD
       CALL FREQCLR(5, 6): CALL FREQCLR(8, 9)
     END IF
     IF ASC(RIGHT$(K$, 1)) = 62 THEN CALL LOAD          '*** [F4]
     IF ASC(RIGHT$(K$, 1)) = 63 THEN                    '*** [F5]
COLOR 6 + 16, 7: OUT DP, 1 + 8                          '*** nco clock off
LOCATE 15, 40 - LEN(MSG$(7)) / 2: PRINT MSG$(7)         '**** Prints message
DO: K$ = INKEY$: LOOP UNTIL K$ = CHR$(0) + CHR$(63)     '*** Wait for "F5" key
COLOR 15, 1: LOCATE 15, 40 - LEN(MSG$(4)) / 2: PRINT MSG$(4)
OUT DP, 1                                               '*** Turns NCO clock ON
     END IF
     IF ASC(RIGHT$(K$, 1)) = 68 THEN GOTO endp          '*** [F6]
END IF
LOOP
endp: COLOR 7, 0: CLS
END             'xxxx END PROGRAM xxxxx END PROGRAM xxxxx END PROGRAM
'-------------------------------------------------------------------------
```

regulator (National Semiconductor equivalent)
D1–D4—1N4148 diode
D5—1N4001 diode
D6—1N4933 fast rectifier diode (Motorola or equivalent)
Q1—2N4401 NPN transistor
Q2—2N4403 PNP transistor

Other components

J1—male DB-25 jack
J2, J3—BNC jack
L1—390 µH choke (Renco No. RL3901 or equivalent)
Miscellaneous: Four-foot length of five-conductor cable, four hex spacers and screws (for PC-board "feet"), clip-on heat sink for TO-220 case, PC board, 9-volt DC adapter, solder.

Note: The following items are available from R.J. Portugal, 52 Susan Lane, North Haven, CT 06473:

- **PC board—$20.00**
- **Parts kit including the PC board board-mounted components (does not include any of the "Miscellaneous" parts listed above or jacks J1–J3)—$64.95**
- **5.25-inch, 1.2 Meg floppy diskette containing QBASIC source program—$12.00**

Please add $3.50 for S&H on continental USA orders. All others add $5.00. Make checks or postal money orders payable to R.J. Portugal.

```
SUB DSPLY
COLOR 15, 10: LOCATE 2, 40 - LEN(MSG$(8)) / 2: PRINT MSG$(8)
COLOR 15, 1: LOCATE 5, 1
PRINT "       [F1]       Enter Frequency 1                                    Hz"
PRINT "                  (0.00 to 10,000,000.00Hz                             usec"
PRINT
PRINT "       [F2]       Enter Frequency 2                                    Hz"
PRINT "                  (0.00 to 10,000,000.00Hz                             usec"
PRINT
PRINT "       [F3]       Clear Frequency 1 and 2 to zero": LOCATE 19, 1
PRINT "       [F4]  Load NCO                          [F5]  Start/Stop PSG output"
PRINT
PRINT "                          (F10)   End program-return to DOS"
COLOR 15, 10
LOCATE 23, 40 - (LEN(MSG$(9)) + LEN(MSG$(10)) / 2: PRINT MSG$(9) + MSG$(10)
COLOR 7, 9: LOCATE 25, 40 - LEN(MSG$(11)) / 2: PRINT MSG$(11); : COLOR 15, 1
END SUB

SUB ERMSG (L1, L2)                          '*** Displays 2 line error message
CALL FREQCLR(L1, L2)
COLOR 14, 4: FOR i = 14 TO 15:      LOCATE i, 9: PRINT MSG$(i - 13); : NEXT i
SLEEP 4: COLOR 15, 1: FOR i = 14 TO 16: LOCATE i, 9: PRINT MSG$(3): NEXT i
END SUB

SUB FREQCLR (L1, L2)            '*** Clears control panel F1 & F2 display areas
COLOR 14, 4: LOCATE L1, 45: PRINT MSG$(4): LOCATE L2, 45: PRINT MSG$(4)
END SUB

SUB FREQUENCY (L1, L2, flg)                 '*** F1 & F2 input routine
COLOR 14, 4: CALL FREQCLR(L1, L2): LOCATE L1, 45: LINE INPUT f$
IF VAL(f$) > 10000000 OR VAL(f$) < 0 THEN CALL ERMSG(L1, L2): GOTO esub
dpf = 0
    FOR i = 1 TO LEN(f$)            '*** F1/2 numeral & multiple decimal point
    x$ = MID$(f$, i, 1)             '*** check
     IF x$ = "." AND dpf = 0 THEN : dpf = 1: GOTO x
     IF x$ = "." AND dpf = 1 THEN CALL ERMSG(L1, L2): GOTO esub
     IF ASC(x$) < 48 OR ASC(x$) > 57 THEN CALL ERMSG(L1, L2): GOTO esub
x:   NEXT i
     f = VAL(f$): LOCATE L1, 45: PRINT USING FMT$; f;  '*** prints freq
LOCATE L2, 45
IF f <> 0 THEN PRINT USING FMT$; 1000000 / f;       '*** period of F1 or F2
IF flg = 1 THEN N1& = K * f             '*** Converts F1 or F2 to a 32bit word
IF flg = 2 THEN N2& = K * f             '*** compatible with NCO input specs
COLOR 22, 7: LOCATE 16, 40 - LEN(MSG$(5)) / 2: PRINT MSG$(5): COLOR 15, 1
esub:
END SUB

SUB LOAD        '*** Converts F1&2 into serial format & shifts them into NCO
FOR j = 3 TO 0 STEP -1: FOR x = 7 TO 0 STEP -1    '***** Xfer first 32 bits
    IF (PEEK(VARPTR(N2&) + j) AND 2 ^ x) > 0 THEN dt = 2 ELSE dt = 0
OUT DP, dt + 4: OUT DP, 1 + dt + 4          '*** Serial data and shift
NEXT x: NEXT j                              '*** pulse generator
FOR j = 3 TO 0 STEP -1: FOR x = 7 TO 0 STEP -1    '***** Xfer second 32 bits
    IF (PEEK(VARPTR(N1&) + j) AND 2 ^ x) > 0 THEN dt = 2 ELSE dt = 0
OUT DP, dt + 4 + 16: OUT DP, 1 + dt + 4 + 16    '*** Serial data and shift
NEXT x: NEXT j                              '*** pulse generator
    OUT DP, 1 + 16                          '*** Xfers new F1&2 to NCO
COLOR 15, 1
LOCATE 16, 10: PRINT MSG$(6)
END SUB
```

emitter-follower (Q1 and Q2), and C8. The final RC section consists of R10, R11, and C4. With the values shown in Fig. 1, the NCO generates a relatively constant 5-volt peak-to-peak signal into an open circuit from 0.1 hertz to 5 megahertz. At 10 megahertz, the output amplitude is down to 3 volts peak-to-peak.

The impedance seen looking into the joined emitters of the two transistors (Q1 and Q2) that make up the simple complimentary emitter-follower output driver is about 11 ohms. The 39-ohm resistor (R17) in series with the output terminal sets the output impedance seen by the load to 50 ohms.

Two operating voltages are required for this circuit to work: +5 volts and −5 volts. Both voltages are derived from a 9-volt AC-to-DC adapter. The +5 volts is supplied by an LM7805 voltage-regulator. A National Semiconductor LM2574 switching regulator generates the −5 volts. Diode D4 in series with the regulators protects the board against accidental power-supply reversals. The +5-volt supply must provide about 200 milliamperes, so the LM7805 should be fitted with a heat sink.

The 32-megahertz oscillator could have been built with discrete components for a few dollars less than the clock module IC4 that is specified. However, 32-megahertz oscillators can be cranky at times, and they tend to be noisy. By contrast, the module specified is small and quite reliable.

Programming

The PC must manipulate four control lines for the PSG to work. The CLOCK line is a PC-simulated shift clock for moving data (on the DATA line) into the NCO buffer. The FLOAD line transfers data from the NCO's input buffer to its active registers, and the ON/OFF line turns the PSG output on and off. The four control lines correspond to the first four lines of the parallel port data register as shown in Table 1.

To program the PSG, first

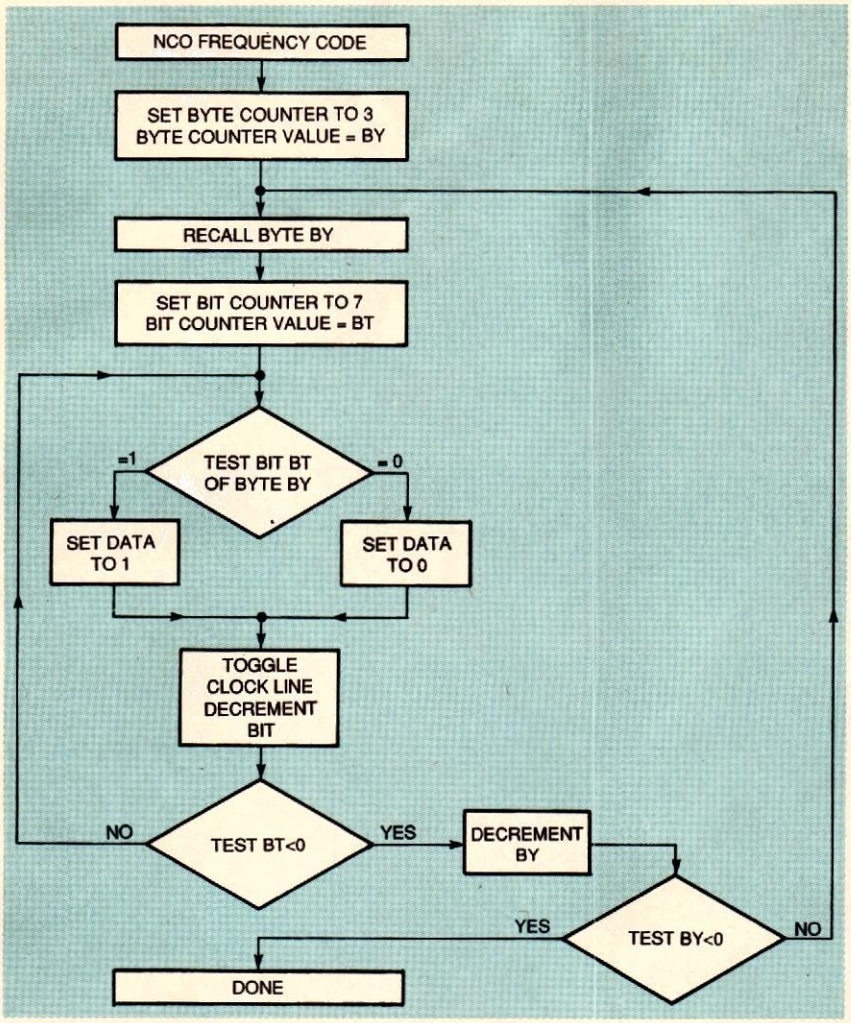


FIG. 2—PARTIAL SUBROUTINE FLOWCHART. The 32-bit NCO frequency code must be read into the board bit-by-bit.

specify some output frequency. Enter that frequency with an input statement (assuming a program written in QBASIC) and then convert the frequency to the format that the NCO wants to see by using the following equation:

N = Integer Portion $(f_{OUT} \times 2^{32}/f_{CLK})$

where: N = NCO code number, f_{OUT} = The output frequency, and f_{CLK} = 32 MHz.

The integer portion of the equation means that only the digits to the left of the decimal point are used for N. Using Basic's long-integer suffix ensures that the NCO frequency code number (N) will be stored in a 32-bit format. An "&" following a variable declares it a long-integer which happens to be 32 bits long.

The program for the PSG shown in Listing 1 is written in QBASIC version 4.5. The program has a few nice features such as: parallel-port selection (LPT1, LPT2, or LPT3), a multicolor display, error messages, and user prompts. The program has been tested on both a 386-based IBM-compatible PC and an old monochrome 8088-based PC running at 4.77 megahertz, and it performs equally well on both machines.

This article will not include a line-by-line analysis of the program. However, consider the "sub load" subroutine. This is where the NCO data word is converted from parallel to serial format and sent to the NCO board. The easiest way to understand what's going on here is to study Fig. 2, a partial flowchart of the subroutine.

First the 32-bit NCO frequency code must be read into the board bit-by-bit. The board is set up for most-significant-bit first entry of the serial data (bit 32 goes to the NCO first and bit 0 is sent last). Next, long-integers in QBASIC are stored as groups of four successive bytes in the PC's memory, least-significant-byte first, most-significant-byte last. Once the memory location of N is known, N must be read in reverse order, fourth byte to first byte. The "Set byte count to 3" and "Decrement BY" lines keep track of which byte is being processed.

The byte now must be scanned from the top down; that is, the most-significant-bit to the least-significant-bit. This is where the "Set bit counter to 7," "Test bit BT," "Set DATA to 1 or 0," "Toggle Clock Decrement BT," and "TEST BT" blocks are used. The program looks at each bit of N, MSB first, and then decides whether the bit is a one or a zero. It then sets the DATA line and toggles the CLOCK line, sending the DATA bit to the NCO.

After each pass through the inside loop (the bit test loop), the bit counter content (BT) is tested to see if all the bits have been scanned—if not, the loop is completed again. When a bit scan has been completed, the BY counter is adjusted and tested to see if the last byte has been processed. Again, the program loops until all bytes have been processed.

With the powerful instructions available in QBASIC, the whole procedure can be completed with just a few lines of code:

```
FOR J = 3 TO 0 STEP-1
FOR X = 7 TO 0 STEP-1
IF PEEK (VARPTR (N&) + J)
AND 2^X > 0 THEN dt = 2 ELSE
  dt = 0
OUT dp, 0 + dt + 4 : OUT dp, 1
  + dt + 4 NEXT J
NEXT X
```

The "IF PEEK..." line is fairly complex. The FOR NEXT J loop accounts for the bytes and the FOR NEXT X loop handles the bit tracking. Starting from the inside and working out, the VARPTR(N&) function returns the address of the variable N&.

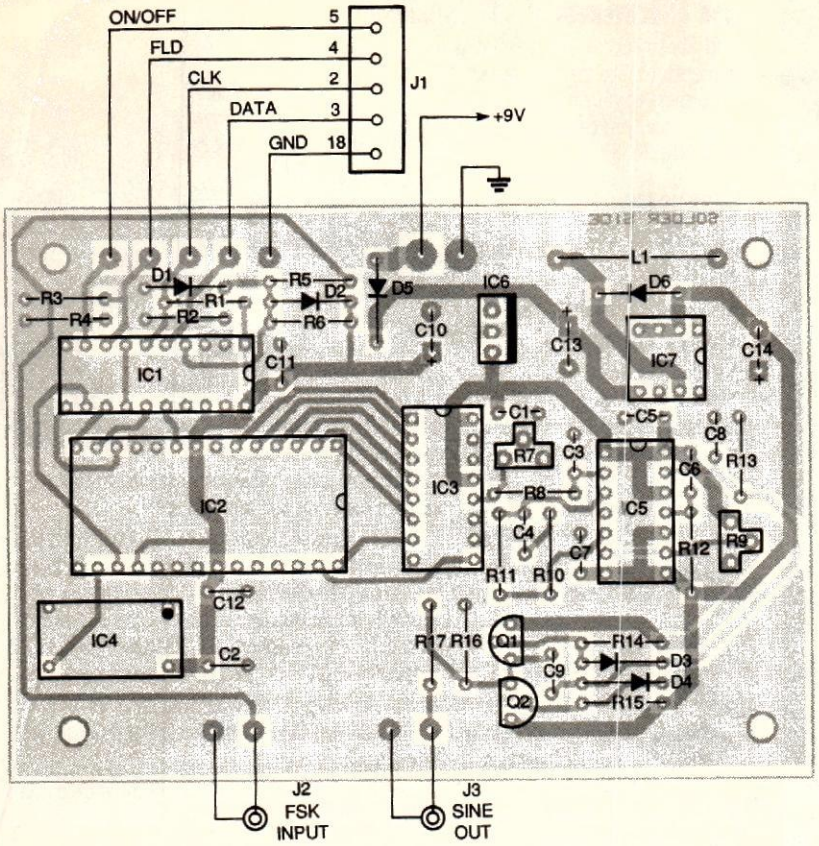


FIG. 3—PARTS-PLACEMENT DIAGRAM. You can make your own board or buy one from the source given in the Parts List. The input/output connectors are not mounted on the board.

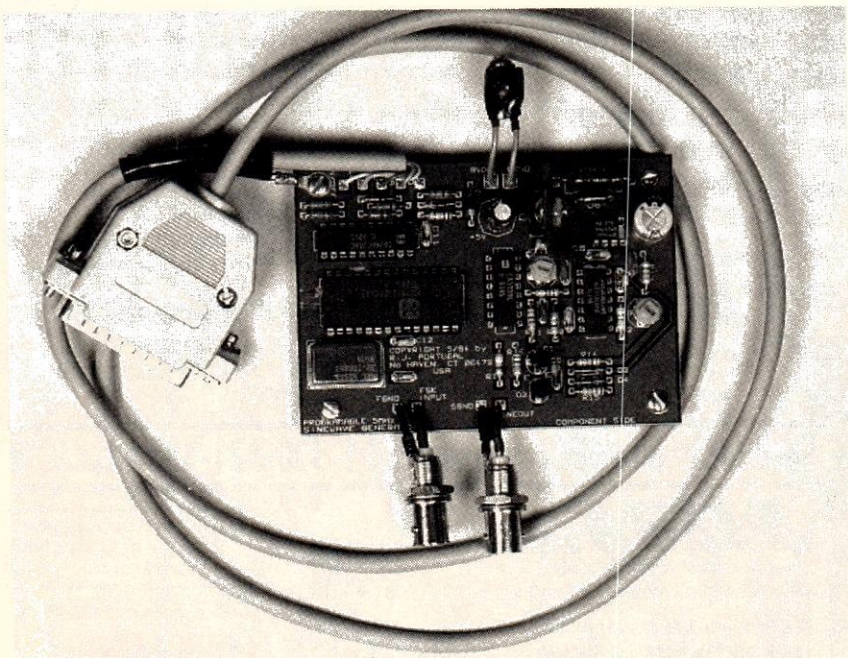


FIG. 4—AN EARLY PSG PROTOTYPE. This board differs slightly in its parts layout from the final design.

that is the address of the first byte of N& (remember that N& consists of four memory bytes). PEEK returns the value of the data byte stored at the specified memory location. So, VARPTR(N&) provides the address of the first byte of N&, and PEEK reads the value of the byte at the address. The "J" inside the PEEK function selects one of the four N& bytes.

The AND logical operator creates the logical results of ANDing two quantities, in this case the eight-bit word supplied by the VARPTR and the eight-bit equivalent of 2^X. Since 2^X takeson binary values only (0, 1, 2, 4, 8, 16, 32, 64 and 128), only one bit of binary word equivalent to 2^X is a high for each value of X. When two bits are ANDed, the result is a one when both bits are ones. If the N& byte being scanned has a one as its MSB, and it is ANDed with 2^7, the result of the AND function will be a binary word equivalent to decimal 128. If the MSB is zero, the resulting word will be zero. This process is called "masking" and it should be familiar to anyone experienced with C or machine-language programming. As the value of X changes from 7 to 0, each bit of the current byte is tested by the AND function.

The final part of the program line, THEN, sets variable dt to either a 2 or a 0. Refer to Table 1. Data to be fed to the NCO appears as the second bit of the parallel-port data word, and it has a binary weight of 2.

The two OUT instructions send bytes to the parallel port data register (dp). The first OUT instruction sends the results of the bit analysis (0 or 2) plus 4 to the NCO. The "4" (refer to Table 1) controls the NCO's FLOAD input that should be set high when data is being transferred to the chip. The second OUT instruction is identical to the first OUT instruction except for an additional "1." That "1" turns on the NCO CLOCK input, producing a positive edge that shifts the value into the NCO.

You might have to read the preceding paragraphs several times so that you understand how the load subroutine works. Once you understood the procedure, you can program the NCO board to do anything it's capable of doing. The rest of the program provides visual enhancements to the screen display, and generally forms a clean control panel for the PSG.

(Continued on page 66)

SINEWAVE GENERATOR

continued from page 52

Construction

A circuit board with a ground plane is required for building this project because of the high frequencies involved. You can make your own board or buy one from the source given in the Parts List. Figure 3 is the parts-placement diagram. The NCO and DAC ICs are expensive and can be damaged if they must be removed, so it is recommended that you install them in low-profile, machined-contact sockets.

Do not mount the input/output connectors on the PC board. BNC connectors are recommended for the FSK input and sinewave output. Attach a grounding lug to the BNC connectors, and then solder it to the ground plane of the board to provide mechanical support for the connectors. Wire the power source directly to the board or wire a suitable power jack to the board.

Cut a four-foot length of five-conductor cable for the PC cable. Solder one end of the cable leads to a male DB-25 connector, as shown in Fig. 3, and the other end directly to the PC board. Figure 4 shows the complete board.

After carefully inspecting the board for incorrectly installed components and poorly soldered joints and making any necessary repairs, the generator is functional. Now you can generate any sinewave you need up to 10 megahertz on the fly. Ω

Go ahead. . .it's an old schematic.

LISTING 1

```
'*************************************************************************
'                          PSG CONTROL PROGRAM                            *
'      COPYRIGHT 5/94 by R.J. PORTUGAL, NORTH HAVEN, CT. 06473, U.S.A     *
'*************************************************************************
DECLARE SUB FREQUENCY (L1, L2, FLAG)        '**** Frequency input routine
DECLARE SUB FREQCLR (L1, L2)                '**** Clear frequency entries
DECLARE SUB LOAD ()                         '**** Parallel-serial data convert
DECLARE SUB ERMSG (L1, L2)                  '**** Error messages
DECLARE SUB DSPLY ()                        '**** Display screen

COMMON SHARED FMT$, N1&, N2&, K, DP, PORT$, MSG$(), L1, L2
DIM Z AS STRING * 1: DIM MSG$(20)
FMT$ = "###,###,###,###.####": Clk = 32000283: K = (2 ^ 32 / Clk)

'*************************************************************************
'       To select NCO parallel printer remove the (') "comment symbol"
'       in front of the appropriate program line.
'       PRINTER PORT "LPT2" will be selected by the following:

     'port$ = "LPT1": dp = &H3BC      '*** LPT1 DATA PORT ADDRESS
      PORT$ = "LPT2": DP = &H378      '*** LPT2 DATA PORT ADDRESS
     'port$ = "LPT3": dp = &H278      '*** LPT3 DATA PORT ADDRESS
'*************************************************************************
'********** (Message list) **********
MSG$(1) = "        Input range is from 0.00Hz to 10,000,000.00Hz          "
MSG$(2) = "     Program accepts numerals and a single decimal point       "
MSG$(3) = STRING$(64, " "): MSG$(4) = STRING$(21, " ")
MSG$(5) = "Last input not transferred to NCO": MSG$(6) = STRING$(50, " ")
MSG$(7) = "PSG output signal OFF"
MSG$(8) = "PROGRAMMABLE SINEWAVE GENERATOR CONTROL SCREEN"
MSG$(9) = "PSG printer port is " + PORT$ + "."
MSG$(10) = PORT$ + " uses I/O port " + HEX$(DP) + "HEX"
MSG$(11) = "COPYRIGHT 5/94 by R.J. PORTUGAL, NORTH HAVEN, CT. 06473, U.S.A."

COLOR 15, 1: CLS : CALL DSPLY: N1& = 0: N2& = 0: CALL LOAD
'-------------------------------------------------------------------------
DO: K$ = INKEY$                                 '*** Main Program
IF LEN(K$) = 2 THEN       'check keyboard entry for 2 character scan code
     IF ASC(RIGHT$(K$, 1)) = 59 THEN CALL FREQUENCY(5, 6, 1)     '**** [F1]
     IF ASC(RIGHT$(K$, 1)) = 60 THEN CALL FREQUENCY(8, 9, 2)     '**** [F2]
     IF ASC(RIGHT$(K$, 1)) = 61 THEN                             '**** [F3]
       N1& = 0: N2& = 0: CALL LOAD
       CALL FREQCLR(5, 6): CALL FREQCLR(8, 9)
     END IF
     IF ASC(RIGHT$(K$, 1)) = 62 THEN CALL LOAD          '*** [F4]
     IF ASC(RIGHT$(K$, 1)) = 63 THEN                    '*** [F5]
COLOR 8 + 16, 7: OUT DP, 1 + 8                          '*** nco clock off
LOCATE 15, 40 - LEN(MSG$(7)) / 2: PRINT MSG$(7)         '*** Prints message
DO: K$ = INKEY$: LOOP UNTIL K$ = CHR$(0) + CHR$(63)     '*** Wait for "F5" key
COLOR 15, 1: LOCATE 15, 40 - LEN(MSG$(4)) / 2: PRINT MSG$(4)
OUT DP, 1                                               '*** Turns NCO clock ON
     END IF
     IF ASC(RIGHT$(K$, 1)) = 68 THEN GOTO endp          '**** [F6]
END IF
LOOP
endp: COLOR 7, 0: CLS
END              'xxxx END PROGRAM xxxxx END PROGRAM xxxxx END PROGRAM
'-------------------------------------------------------------------------
```

```
SUB DSPLY
COLOR 15, 10: LOCATE 2, 40 - LEN(MSG$(8)) / 2: PRINT MSG$(8)
COLOR 15, 1: LOCATE 5, 1
PRINT "       [F1]       Enter Frequency 1                                    Hz"
PRINT "                  (0.00 to 10,000,000.00Hz                             usec"
PRINT
PRINT "       [F2]       Enter Frequency 2                                    Hz"
PRINT "                  (0.00 to 10,000,000.00Hz                             usec"
PRINT
PRINT "       [F3]       Clear Frequency 1 and 2 to zero": LOCATE 19, 1
PRINT "       [F4]  Load NCO                          [F5]  Start/Stop PSG output"
PRINT
PRINT "                          (F10)   End program-return to DOS"
COLOR 15, 10
LOCATE 23, 40 - (LEN(MSG$(9)) + LEN(MSG$(10)) / 2: PRINT MSG$(9) + MSG$(10)
COLOR 7, 9: LOCATE 25, 40 - LEN(MSG$(11)) / 2: PRINT MSG$(11); : COLOR 15, 1
END SUB

SUB ERMSG (L1, L2)                    '*** Displays 2 line error message
CALL FREQCLR(L1, L2)
COLOR 14, 4:  FOR i = 14 TO 15:       LOCATE i, 9: PRINT MSG$(i - 13); : NEXT i
SLEEP 4: COLOR 15, 1: FOR i = 14 TO 16: LOCATE i, 9: PRINT MSG$(3): NEXT i
END SUB

SUB FREQCLR (L1, L2)            '*** Clears control panel F1 & F2 display areas
COLOR 14, 4: LOCATE L1, 45: PRINT MSG$(4); LOCATE L2, 45: PRINT MSG$(4)
END SUB

SUB FREQUENCY (L1, L2, flg)                   '*** F1 & F2 input routine
COLOR 14, 4: CALL FREQCLR(L1, L2): LOCATE L1, 45: LINE INPUT f$
IF VAL(f$) > 10000000 OR VAL(f$) < 0 THEN CALL ERMSG(L1, L2): GOTO esub
dpf = 0
     FOR i = 1 TO LEN(f$)             '*** F1/2 numeral & multiple decimal point
     x$ = MID$(f$, i, 1)              '*** check
      IF x$ = "." AND dpf = 0 THEN : dpf = 1: GOTO x
      IF x$ = "." AND dpf = 1 THEN CALL ERMSG(L1, L2): GOTO esub
      IF ASC(x$) < 48 OR ASC(x$) > 57 THEN CALL ERMSG(L1, L2): GOTO esub
x:   NEXT i
     f = VAL(f$): LOCATE L1, 45: PRINT USING FMT$; f;  '*** prints freq
LOCATE L2, 45
IF f <> 0 THEN PRINT USING FMT$; 1000000 / f;          '*** period of F1 or F2
IF flg = 1 THEN N1& = K * f                '*** Converts F1 or F2 to a 32bit word
IF flg = 2 THEN N2& = K * f                '*** compatible with NCO input specs
COLOR 22, 7: LOCATE 16, 40 - LEN(MSG$(5)) / 2: PRINT MSG$(5): COLOR 15, 1
esub:
END SUB

SUB LOAD       '*** Converts F1&2 into serial format & shifts them into NCO
FOR j = 3 TO 0 STEP -1: FOR x = 7 TO 0 STEP -1     '***** Xfer first 32 bits
     IF (PEEK(VARPTR(N2&) + j) AND 2 ^ x) > 0 THEN dt = 2 ELSE dt = 0
OUT DP, dt + 4: OUT DP, 1 + dt + 4                 '*** Serial data and shift
NEXT x: NEXT j                                     '*** pulse generator
FOR j = 3 TO 0 STEP -1: FOR x = 7 TO 0 STEP -1     '***** Xfer second 32 bits
     IF (PEEK(VARPTR(N1&) + j) AND 2 ^ x) > 0 THEN dt = 2 ELSE dt = 0
OUT DP, dt + 4 + 16: OUT DP, 1 + dt + 4 + 16       '*** Serial data and shift
NEXT x: NEXT j                                     '*** pulse generator
     OUT DP, 1 + 16                                '*** Xfers new F1&2 to NCO
COLOR 15, 1
LOCATE 16, 10: PRINT MSG$(6)
END SUB
```