By George L. Heron, N2APB

PSK31 Audio Beacon

Build this programmable single-chip generator of PSK31-encoded audio data streams and use it as a signal generator, a beacon input to your SSB rig—or as the start of a single-chip PSK31 controller!

Here's an easy, fun and intriguingly useful project that has evolved from an ongoing design effort to reduce the complexity of a PSK31 controller.

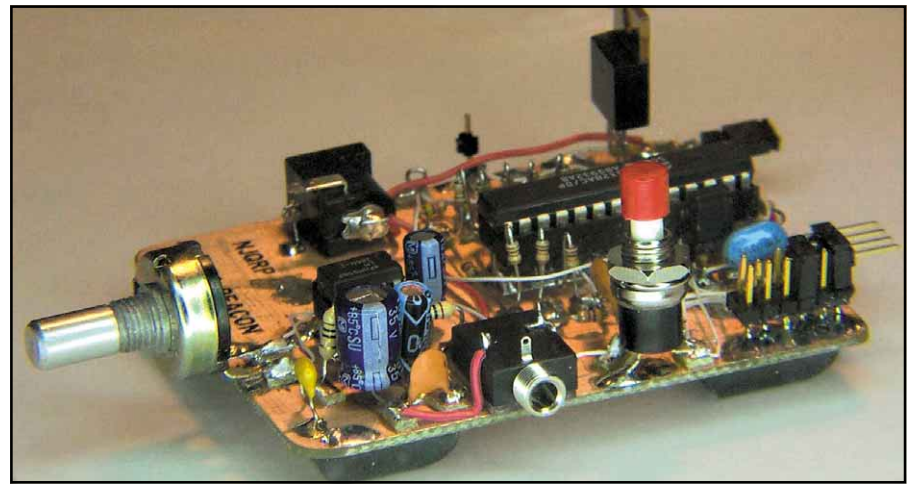
A conventional PC typically provides the relatively intensive computing power required for PSK31 modulation and demodulation. With this beacon project, however, the PSK31 modulation computations have been designed to fit into a small PIC-like microcontroller that can serve as the basis for the "transmit half" of a standalone PSK31 controller.

A fast and inexpensive microcontroller is programmed to generate an audio data stream using the PSK31 algorithms. The data-driven audio waveform is fed to an amplifier IC that drives a speaker and *voila*, the familiar and melodious PSK31 warble is heard! When presented as input to a PSK31 receiving system such as *DigiPan*, these modulated audio tones are decoded and the programmed beacon string is displayed.

A keyboard or data terminal may also serve as the input of real-time textual data to the PSK31 audio beacon. A standard RS-232 serial interface is provided in the hardware and software to allow a more dynamic "signal generator" use of the project. The project can be electrically connected to the input of an SSB transceiver to create an RF PSK31 beacon for brief tests.

This project is also ideally suited for groups wishing to have some "audio beacon" fun during meetings. A number of club members would operate their audio beacons while someone attempts successful copy of the beacon strings while sitting at a microphone-equipped laptop running *DigiPan* software.

Construction is simple and straightforward, and you'll have immediate feedback on how your beacon works when you plug in a 9-V battery and speaker.



Beacon Features

- Single-chip implementation of PSK31 encoding and audio waveform generation.
- An on-board audio amplifier sufficient to drive a speaker for group activities.
- A low-level output suitable for interfacing to an SSB transceiver.
- A Scenix SX28 RISC microcontroller operating at 50 MHz with a 20-ns instruction cycle time. This provides computing power necessary for accurate implementation of the PSK31 modulation algorithm. The SX chip is similar to Microchip's popular PIC microcontroller, containing the same software instruction set but operating more than 40 times faster.
- The SX28 microcontroller is programmed with a unique beacon string. It can also accept real-time text input from an RS-232 serial interface.
- Configuration jumpers provide for selection of three base carrier frequencies (500 Hz, 1 kHz or 2 kHz), and choice of 16 sub-variations around the selected base frequency. This allows the user to operate the beacon on any of 48 distinct audio frequencies.
- Continuous loop or single-pass operation.
- Open source code for custom modification of the beacon string and/or software operation.
- Construction may be done Manhattan-style (a form of ugly-style construction) for freedom of desired implementation. A printed circuit board is also available for this project.

Typical Beacon String

The current version of the beacon software is programmed to transmit two types of data in sequence:

(1) Idle Stream—Upon transmit initiation, the beacon sends a series of 64 zeros to allow the PSK31 receiving system and decoder to synchronize for the data reception that follows. In some PSK31 applications this idle stream time allows the decoding software to measure signal "IMD," an indication of energy present in adjacent sidebands and somewhat of a figure of merit for the received signal.

(2) Data String—Immediately following the idle stream of zeros, the beacon begins sending the data string that will ultimately be displayed on the receiving side of the communications channel. This is the custom-programmed sequence

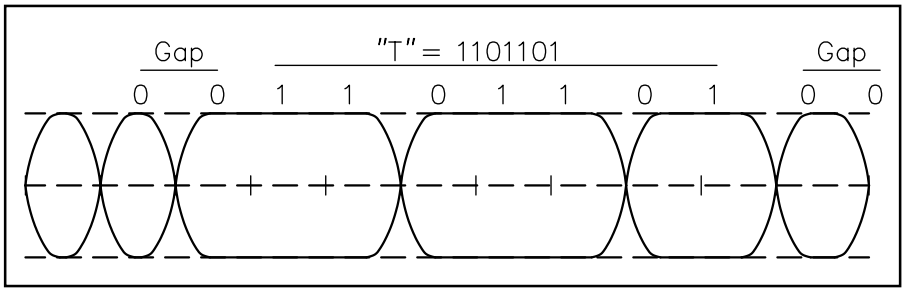


Figure 1—This represents 15 cycles of the 500-Hz carrier constituting a 31-ms bit-processing window. The sequence of characters processed were two sequential zeros—the first being encoded in the phase reversal seen at the zero-power point on the left, and the second on the right.

of ASCII characters. The data string may be of any length, and is limited only by available memory.

What Can You Do with a PSK31 Audio Beacon?

As I mentioned, the beacon may be used as the basis for a fun group "contest" activity for your club. All contestants would turn their beacons on and gather around a laptop running *DigiPan* software. Laptops generally have built-in microphones that would be used to decode the audio PSK31 tones "in the air" ... and with over 100 beacons warbling simultaneously, there will certainly be audio tones in the air!

Recall that each beacon's microcontroller can be customized with a call sign and can also operate on slightly different frequencies. One beacon may have its tones centered at 978 Hz, while another may have its tones at 1050 Hz.

Imagine a bunch of club members amassed around a table with an operator sitting at the laptop. The idea is to see how many beacons the operator can copy within a specified period of time. Factors involved in successful reception include the settings of the audio amp, the type of speaker, the distance between the beacon and the laptop, adjacent QRM (other beaconers) and so on. "Points" are awarded to all beacons for the degree of solid copy captured during the time period. (You can see a photo of this excitement in the "Up Front" section of July *QST*. The photo was taken at the Atlanticon QRP convention in Maryland last spring.)

This all may sound complex, but it's really quite simple—build the beacon, turn it on and see how well it can be copied by *DigiPan*. Each contestant could actually do this during the test phase at home prior to the club event.

Putting the Beacon on the Air

Projects can be fun in a group setting, but the PSK31 audio beacon has lasting value for the individual PSK31 enthusiast. The audio tones generated by the beacon can be used to drive any SSB transceiver. This would allow you to easily put a beacon signal on the air for all the same reasons that CW beacons are used (studies of propagation, power levels, antenna characteristics, etc).

Special care must be exercised on two counts, however.

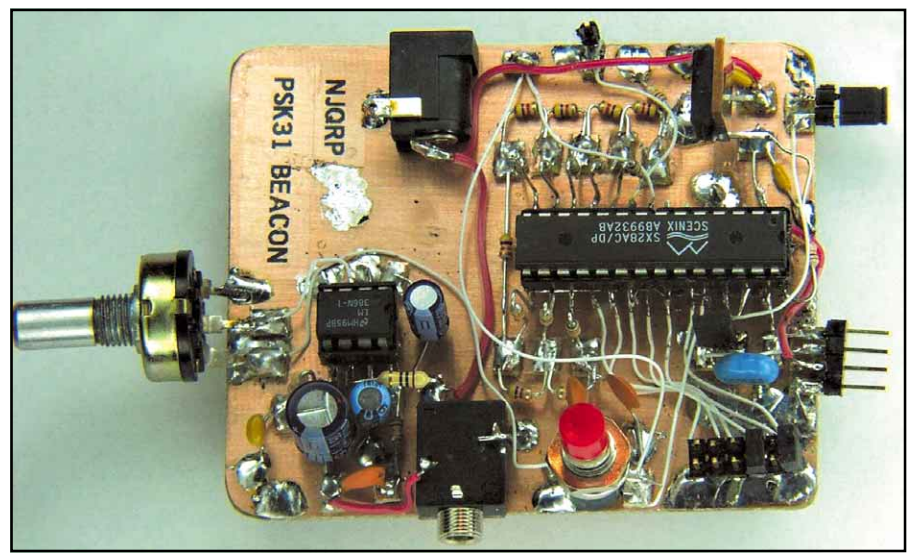
First, the audio level driving an SSB transmitter *must be extremely low* compared to the speaker output levels provided by the beacon kit. Most certainly, the LM386 audio amplifier output that drives the beacon's speaker should *not* be used when feeding a transmitter. You should take the output of the R-2R DAC and put it through a voltage divider pad (or potentiometer) to bring the 0-to-4.5-V sine wave signals down to the millivolt range required by an SSB transceiver. The lower the better! If the transmitter is overdriven, all sorts of problems occur with the transmitted RF spectrum—terrible intermodulation distortion (IMD) and interference to other signals up and down the band. You will quickly earn the wrath of others who will see your call sign and e-mail address transmitted over and over.

Secondly, even in the case of perfect signal quality, *never leave the beacon on for continuous, unattended operation*. Not only would this be illegal in most instances, it is also very poor amateur practice. This beacon is intended only for *brief* testing when used over the airwaves.

Circuit Description

Refer to the schematic shown in Figure 3. The kit provides a standard battery clip with which the user can connect a 9-V battery. Any dc voltage from about 9-12 V may be used since the 3-pin regulator VR1 drops the input voltage down to the required 5 V for the microcontroller. Current consumption of the beacon circuitry is nominally about 80 mA, so the regulator will naturally get a little warm. If an SX-Key programmer is connected to the beacon (allowing reprogrammability of the microcontroller), a TO220-packaged 1-A voltage regulator such as an LM7805 should be substituted due to the overall higher current demands of the programmer.

The Scenix SX28 microcontroller used in the beacon operates at 50 MHz clock rate, providing an instruction cycle time of 20 nanoseconds. This fast operation enables precise control of signal generation and phase reversals to produce stable and accurate carrier modulation at the audio baseband frequencies. A 50-MHz ceramic resonator is used with



Notice the potentiometer used for audio drive control into the LM386, the 1/8-inch jack used for the speaker output, the pin headers used for configuration, the coaxial dc power jack used for the dc input power, the real START pushbutton, and the 4-position pin header on the right used for the SX-Key programmer connection. This little board has lots of flexibility and is relatively self-contained.

the on-board oscillator to provide a fast and simple controller solution to the generation of PSK31 encoding.

There are numerous ways to generate a sine wave suitable for use in communications systems, and each has advantages and trade-offs. A discrete chip sine wave generator or a separate digital to analog converter (DAC) could have been used, but it was desired to keep both hardware complexity and cost to a minimum.

Another popular method used in generating a sine wave is to pulse width modulate (PWM) a square wave on an output bit of the microcontroller and then low pass filter the signal with an R/C network. This method requires too much use of precious interrupt time in the processor.

A simple technique was ultimately chosen to generate the carrier—the R-2R DAC. This digital-to-analog converter incorporates a ladder network of 16 resistors whose nodes are fed by an 8-bit parallel output port of the microcontroller. The values of the resistors in the network are 10 kΩ and 20 kΩ; hence the R-2R nomenclature. The cumulative weighting of these R-2R resistors in the ladder ultimately produces an output voltage at the top of the ladder corresponding to the desired analog voltage. Thus all the software needs to do is present the desired sine wave values in sequence to the output port at precise time intervals. When smoothed with a capacitor, the resultant waveform at the top of the resistive ladder is a clean sine wave.

The output of the R-2R DAC is ac coupled to the input of a common LM386 audio amplifier through a potentiometer to provide continuous adjustability of the audio volume.

The beacon is flexible thanks to the use of seven configuration jumpers that instruct the software to produce one of 48 distinct carrier frequencies that will ultimately be phase-modulated at 31 baud. Seven input pins of the SX microcontroller are used to read the status of these configuration jumpers. These input pins have weak internal pull-up resistors and float "high" when unconnected. However, when grounded by putting a jumper in place, the pin reads "low" and signals the software to take specific action in configuring the beacon's frequency and operation.

Carrier Frequency Selection

- Base Carrier Selection Jumpers—Two jumpers allow user to configure beacon carrier signal to any of three frequencies: 500 Hz, 1 kHz or 2 kHz.
- Carrier Offset Jumpers—Four additional jumpers allow user to select one of 16 closely spaced frequencies around the chosen base carrier.

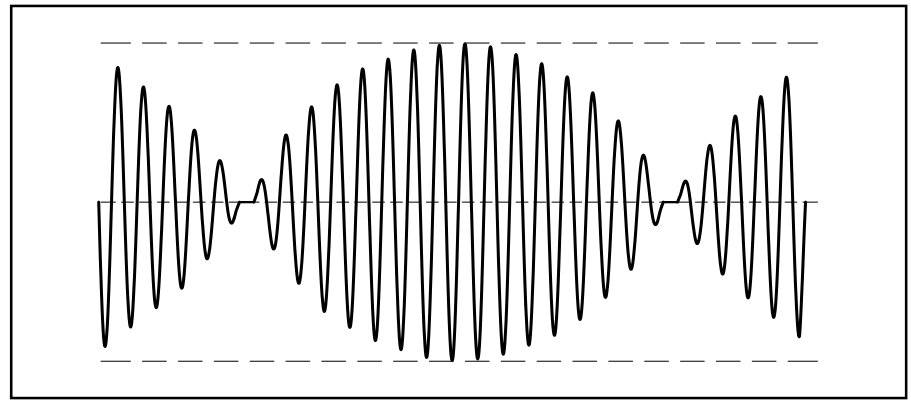


Figure 2—This is an illustration of what you'd probably see with an oscilloscope properly synchronized at the start of the PSK31 character sequence. Note the phase-shifts at the zero-power points, indicating a letter gap of two successive zeroes in the bitstream.

Transmission Mode

- Continuous transmit is selected by installing configuration jumper X5, thus instructing the beacon software to automatically restart the beacon transmit sequence (idle and pre-programmed data string).
- Single pass (one time) transmit is selected by removing the configuration jumper from X5. The transmit sequence is initiated by manual actuation of the **START** pushbutton, upon which the idle stream and pre-programmed data string are sent. The beacon stops transmitting at the end of the data string and awaits either another **START** pushbutton actuation.

Firmware Description

For this part of the discussion, we'll assume that the configuration jumpers are set to produce a 500-Hz carrier frequency, with a nominal interrupt variability of 4 (X2 jumper in place for an RTCC reload value of 197).

The PSK31 audio beacon firmware is completely interrupt-driven, based on the timeout settings of the real time clock (RTCC). The default setting of the RTCC counter produces an interrupt every 3.94 µs and the Interrupt Service Routine (ISR) counts two of these interrupts and then signals the presence of a 7.88 µs interval by setting the SYNC7US software flag. This flag is inspected in a tight loop at the main starting point of the program, and when detected as being set, the whole program sequence begins.

Every fourth time the 7.88 µs window

Bryan Williams, AA3WM, built his PSK31 Beacon kit into the shell of a flashlight, providing ultimate portability!

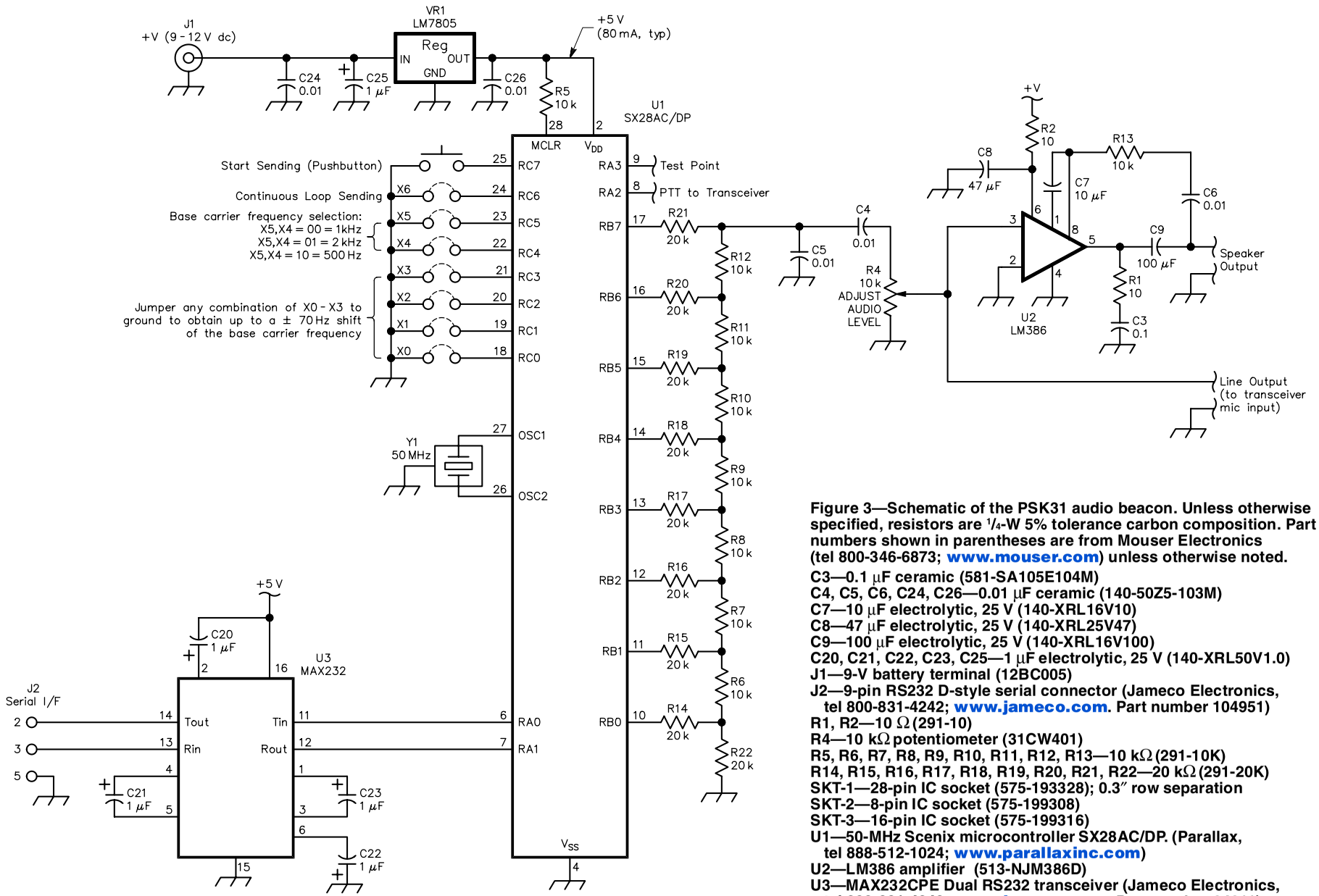


Figure 3—Schematic of the PSK31 audio beacon. Unless otherwise specified, resistors are 1/4-W 5% tolerance carbon composition. Part numbers shown in parentheses are from Mouser Electronics (tel 800-346-6873; www.mouser.com) unless otherwise noted.

C3—0.1 µF ceramic (581-SA105E104M)
C4, C5, C6, C24, C26—0.01 µF ceramic (140-50Z5-103M)
C7—10 µF electrolytic, 25 V (140-XRL16V10)
C8—47 µF electrolytic, 25 V (140-XRL25V47)
C9—100 µF electrolytic, 25 V (140-XRL16V100)
C20, C21, C22, C23, C25—1 µF electrolytic, 25 V (140-XRL50V1.0)
J1—9-V battery terminal (12BC005)
J2—9-pin RS232 D-style serial connector (Jameco Electronics, tel 800-831-4242; www.jameco.com. Part number 104951)
R1, R2—10 Ω (291-10)
R4—10 kΩ potentiometer (31CW401)
R5, R6, R7, R8, R9, R10, R11, R12, R13—10 kΩ (291-10K)
R14, R15, R16, R17, R18, R19, R20, R21, R22—20 kΩ (291-20K)
SKT-1—28-pin IC socket (575-193328); 0.3″ row separation
SKT-2—8-pin IC socket (575-199308)
SKT-3—16-pin IC socket (575-199316)
U1—50-MHz Scenix microcontroller SX28AC/DP. (Parallax, tel 888-512-1024; www.parallaxinc.com)
U2—LM386 amplifier (513-NJM386D)
U3—MAX232CPE Dual RS232 transceiver (Jameco Electronics, tel 800-831-4242; www.jameco.com. Part number 24811)
VR1—5-V voltage regulator, 1 A (511-L7805 ACV).
Y1—50-MHz Murata 250-05060 ceramic resonator (Parallax, tel 888-512-1024; www.parallaxinc.com)

starts (i.e., at the 31.25 µs boundaries), the software gets the next 8-bit value from the sine wave look-up table and outputs it to the DAC output port RB. When this process happens 64 times (i.e., when 64 instances of the 31.25-µs windows have occurred) a single sine wave will have been constructed within a 2-µs time period, creating a 500-Hz carrier.

The software keeps track of how many carrier cycles have been generated, and when the count reaches 15 (i.e., at the 31-ms interval), the PSK bit window is present and the PSK bit processing begins.

Actually, the recurring 31-ms window starts at mid-position of one bit cycle and goes to mid-position of the next bit cycle. It's done this way so the software can inspect the current/next bit relationship and command a zero-power phase reversal condition at the next end-bit time period, if required.

The bits constituting the PSK31 Varicode character being sequentially presented for modulation are inspected on a bit-by-bit basis at each 31ms bit-processing window. When a "1" is encountered, nothing is done in that window. The sine wave construction continues

However, when a "0" is encountered, the rules of PSK31 modulation state that a phase reversal must be forced in the carrier.

The processing gets a little more complicated at this point because we want to reduce the power of the carrier at the time of phase reversals. This action greatly reduces the glitch energy at the time of the reversals and makes the resultant tones much more spectrally clean.

A cosine wave look-up table is used to modulate (or "scale") the carrier wave, sample by sample at the 31.25-µs rate. The cosine table pointer is advanced as each cycle of the carrier is generated.

If the "next bit" of the Varicode character to be processed is a "0," the scaling process is turned on at mid-bit position and the remaining seven carrier cycles of waveform construction occurring in the bit-processing window get scaled per the cosine look-up table. This effectively brings the amplitude of those seven sine waves progressively down to zero, at which point the phase of the carrier is reversed (by changing to a different sine wave look-up table—one that is 180 degrees out of phase from the other one).

After the phase reversal, the carrier continues to be constructed at every 31.25-µs interval, and the cosine scaling is still engaged in sync with the carrier cycles. For the next seven cycles, the cosine look-up table routines scale up the carrier such that the carrier is back to full power (no scaling) by mid-bit position. All this can be more easily understood by considering the representation shown in Figure 2.

In Figure 1 we see 15 cycles of the 500-Hz carrier constituting a 31-ms bit-processing window. The sequence of characters processed were two sequential zeros—the first being encoded in the phase reversal seen at the zero-power point on the left, and the second on the right. These points are where the sine wave look-up tables are changed, resulting in the obvious phase reversals. It can also be seen that the cosine "scaling" of the carrier starts at mid position in the bit-processing window and proceeds to zero at the end of the cycle where the reversals occur. The power is then raised in the following seven cycles of the next window.

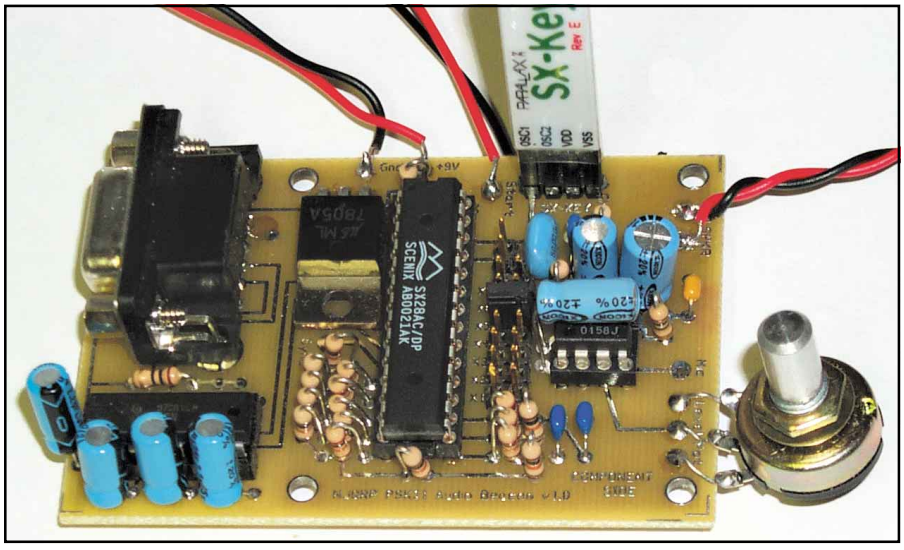
Using the PSK31 encoding algorithm for phase reversals (i.e., insert a phase reversal whenever encountering a "0" in the bit stream), we can begin to consider what a composite sequence of bits will look like for an entire character. In this example, we'll look at the letter "T", which has a Varicode equivalent bit pattern of 1101101. The illustration shown in Fig-

References

- Scenix programming tools, manuals, parts and other technical information:
  Parallax—**www.parallaxinc.com**
  Ubicom (formerly Scenix)—**www.ubicom.com**
  SXTECH—**www.sxtech.com**
  SX Forum—**www.sx-forum.com**
- PSK31 audio beacon kit Web site, containing the source code, manual revisions, kit notes, tips and techniques—**www/njqrp.org/psk31beacon/psk31beacon.html**. The source code is also available on *ARRLWeb* at **www.arrl.org/files/qst-binaries/** as audiopskv1.txt.
- A parts kit for the PSK31 audio beacon, including a printed circuit board, may be ordered from the NJQRP Club for $25. DX orders please add $3 for shipping. When ordering, please specify the desired text string (50 characters, maximum) for pre-programming of SX chip. Suggestion: call sign and e-mail address.

  Checks and money orders must be payable to:
  George Heron, N2APB
  2419 Feather Mae Court
  Forest Hill, MD 21050

  PayPal electronic payments are also accepted (specify the payment as being made to **n2apb@amsat.org**).
- See the PSK31 Web site for a robust listing of PSK31-related articles, technology descriptions and more—**www.psk31.com**.



Here's one of the prototype PSK31 Beacon printed circuit boards in the process of being programmed with the "SX Key" cable connected to the development computer. Note the DB9 serial connector and MAX232 driver chip at the left of the board providing serial communication with the PC during Beacon use. An electrolytic capacitor is mounted in an unusual manner over the LM386 audio amp on the right—the final layout of the pcb brings this cap down to the board, as well as providing for a board-mounted audio-level potentiometer.

It's useful at this point to mention that two (or more) consecutive "zeros" constitute a letter gap, thus instructing the decoding engine on the receiving side to start another character processing cycle.

ure 1 represents what you'd probably see with an oscilloscope properly synchronized at the start of the character sequence.

It's useful at this point to mention that two (or more) consecutive "zeros" constitute a letter gap, thus instructing the decoding engine on the receiving side to start another character processing cycle.

In Figure 1, a letter gap of 0-0 starts off the "T" sequence of bits. Upon encountering each "0," the power level is brought down to zero and the phase is reversed. The power level is then raised back up to 100% by mid-next position, whereupon the "next bit" is inspected to see if another reversal will be needed.

Upon encountering the first "1", the algorithm dictates that no phase reversal occurs, so the power level remains at 100% (no cosine scaling). The same happens again upon encountering the second "1", but the next bit after that is a "0" and the power level is reduced in anticipation of the coming phase reversal.

Producing Different Frequency Carriers

The preceding discussion assumed a 500 Hz carrier and a core interrupt time of 3.94 µs. With a little software magic, some variability was placed into the program in order to produce two more base carrier frequencies, and 16 sub-variations around each base frequency.

The interrupt structure and timing is actually designed to produce a 2-kHz carrier as the highest base frequency, using a 7.88 µs sine wave construction. Then, based on the state of the configuration jumpers, we're able to sample at half-rates to get the 15.75-µs rate for 1 kHz carrier generation, and the 31.25 µs rate for 500-Hz carrier generation.

Achieving the 16 frequency variations around each of the three base carrier frequencies is accomplished by slightly varying the basic underlying interrupt timing mechanism.

The master 50-MHz oscillator internal to the microcontroller clocks the Real Time Clock Counter (RTCC). Interrupts are generated when the RTCC "rolls over" upon a countdown from a preset value, and the governing mechanism for interrupt timing is to preset the RTCC counter at the end of each interrupt cycle. The nominal value of the RTCC preset is "200", and we can go as much as ±8 counts before receiving systems lose synchronization with the master 31 baud system timing in the beacon.

Therefore, the RTCC variable is set to a value between 194 and 209 within the setRTCC routine based on the state of the configuration jumpers X0 through X3. These four bits give 16 different RTCC preset values that modify the basic system timing of the beacon, which results in an ability to more precisely position the base carriers within about ±60 Hz around their nominal values.

The last comment about the software design is that I make extensive use of the look-up table (LUT) capabilities of the SX microcontroller. Using the LUTs, we are able to easily generate two 64-point sine waves—a positive-going one SINETBP and a negative-going one (SINETBN)—and a 64-point half-cosine waveform (COSTBL). Using a pointer to travel through each table allows easy retrieval of the waveform values, which represent a 0-to-1 percentage of the 5 bit values represented in the tables.

Building the Beacon

There's really not too much to assembling this project ugly style—it's really just a couple of ICs and a handful of resistors and capacitors mounted on a small card. All you'll need to do is to assemble the circuits according to the schematic in Figure 3, plug the chips into the sockets, connect a dc power source and a speaker and the beacon should work. No alignment, no muss, no fuss!

The parts are all commonly available. If you don't want to track down your own components, you can buy the PSK31 audio beacon as a kit from the NJQRP Club. The kit includes a microcontroller preprogrammed with your call sign and e-mail address (if desired). If you want to load and configure the microcontroller yourself, the source code is also available from NJQRP. See the "References" sidebar.

Carefully inspect the wire connections, solder joint quality and component placement. Before inserting the ICs into their sockets, apply dc power and check that 5 V is present on pin 2 of the U1 socket. Also make sure that 9 V is present on pin 6 of the U2 socket.

Remove the dc power and carefully insert the ICs to their respective sockets, ensuring proper orientation of pin 1 of each IC. Apply dc power again and depress the **START** pushbutton (or place the **CONTINUOUS** jumper in place). You should hear a relatively loud PSK31 warble tone coming from the speaker for the duration of the beacon transmission. Volume may be lowered by adjusting the potentiometer. The acid test of your success will be to power up a computer running *DigiPan* or another PSK31 program and present the beacon audio as input to the computer. Many computers and most laptops have microphones that allow the tones to be "heard" by the program. Your beacon's warble should be visible on or around one of the base frequencies on the software display: 500 Hz, 1 kHz or 2 kHz. Place the cursor on that displayed signal and you should see your beacon's programmed sequence displayed in the text portion of the display. Experiment with the jumpers to see and hear the flexibility available in your beacon's frequency settings.

In Case of Trouble

If you had any problems in the checkout step, you should first recheck for proper voltages, proper IC orientations in the sockets and good solder joints. When the beacon is running, the Test Point will be a continuous 16 Hz square wave (31 ms high, 31 ms low) that can be seen as a voltage bouncing around between 2-to-3 V on a dc voltmeter. If you have an oscilloscope, you should see the PSK31 waveforms at the output of the DAC. You will also see a relatively constant 2.4-V reading at this same point when using a dc voltmeter. The PTT pad will be at 5 V during transmission and at 0 V when the beacon is stopped. Current consumption of the beacon is nominally about 80 mA.

Summary

Even if you haven't yet made the plunge into on-the-air PSK31 operation, this project gets you *in* the air with audio warbles that can be used for club fun as well as for test and alignment purposes on the workbench. As this project evolves, I hope to add a single-chip demodulator design to serve as a companion to this beacon modulator chip, which will then serve as a complete modem for a standalone PSK31 controller. *Special thanks are due to our club technical advisors Joe Everhart, N2CX and Dave Benson, NN1G, for their generous support during the development of this project.*

You can contact the author at 2419 Feather Mae Court, Forest Hill, MD 21050; **n2apb@amsat.org**.