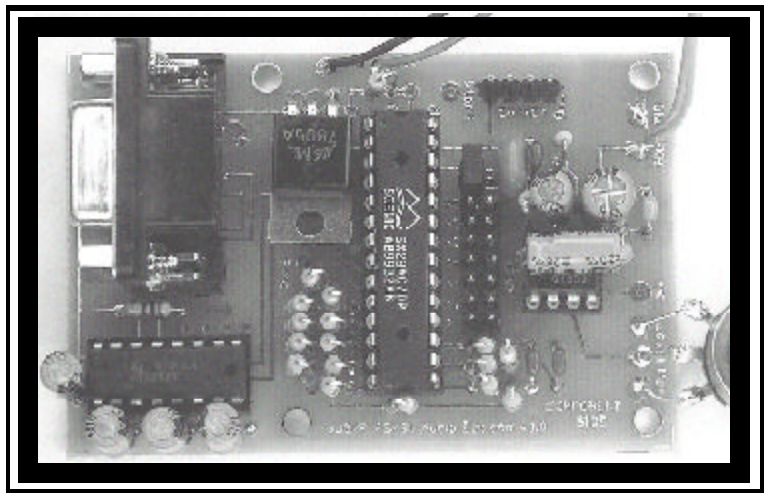


PSK31 Audio Beacon Kit

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!

Version 2, July 2001

Created by the New Jersey QRP Club

PSK31 Audio Beacon Kit

Thank you for purchasing the PSK31 Zudio Beacon from the New Jersey QRP Club! We think you'll have fun assembling and operating this inexpensive-yet-flexible audio modulator as PSK31-encoded data streams.

This project was first introduced as a kit at the Atlanticon QRP Forum in March 2001, and then published in a feature article of QST magazine in its August 2001 issue. Since these initial public exposures, several new capabilities have been added in "version 2" of the microcontroller software. This kit embodies this new software and describes how to use the new features.

And lastly, there were a few typographical errors in the QST article that have been corrected in this kit. Most notably, the graphics depicting the PSK31 waveform encoding now properly indicate waveform phase relationships at bit transitions, and the schematic has been augmented with helpful notations.

For the very latest information, software updates, and kit assembly & usage tips, please be sure to see the PSK31 Beacon website at http://www.njqrp.org/psk31beacon/psk31beacon.html . Visit soon and often, as it's guaranteed to be helpful!

We really hope you find this Manhattan-style Homebrewing Starter Kit useful in getting started in your first copper-clad board project. It's a fun, quick and flexible way to bring your favorite designs to life!

George Heron, N2APB
email: n2apb@amsat.org
for the NJQRP Club
at http://www.njqrp.org

PARTS LIST

U1	SX28AC/DP Scenix microcontroller, 50 MHz	
U2	LM386 amplifier	
U3	MAX232CPE Dual RS232 transceiver	
Y1	50 MHz ceramic resonator, Murata 250-05060	
SKT-1	28-pin IC socket	
SKT-2	8-pin IC socket	
SKT-3	16-pin IC socket	
VR1	7805V voltage regulator, 1A	
C3	0.1 uF ceramic ("104")	
C4,C5,C6,C24,C26	.01 uF ceramic ("103")	
C20,C21,C22,C23,C25	1 uF electrolytic, 25V (longer lead is positive)	
C7	10 uF electrolytic, 25V (longer lead is positive)	
C8	47 uF electrolytic, 25V (longer lead is positive)	
C9	100 uf electrolytic, 25V (longer lead is positive)	
R1,R2	10-ohm	
R4	10K potentiometer	
R5,R6,R7.R8,R9,R10,R11,R12,R13		10K
R14,R15,R16,R17,R18,R19,R20,R21,R22		20K
R3,R4	100K	
R23	470K	
J1	9V battery terminal clip	
J2	9-pin RS232 D-style serial connector	
PC Board		

OVERVIEW

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 has been 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 able to be 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 transmitter to create an RF PSK31 beacon.

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 laptop equipped with DigiPan software.

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

Beacon Features

- Single-chip implementation of PSK31 encoding and audio waveform generation.
- Audio PSK31 tones presented to an audio amp for use with a user-supplied speaker for use in group activities.
- Lower level signal available from chip suitable for input to an SSB transmitter for operation as an RF beacon.
- Scenix SX28 RISC microcontroller operating at 50 MHz with 20ns instruction cycle time 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 over 40 times faster.
- SX28 microcontroller is programmed with a unique beacon string, or accepts 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 8 sub-variations around the selected base frequency. This allows the user to operate the Beacon on any of 24 distinct audio frequencies.
- Configuration jumper provides for Beacon operation in a continuous loop or as a single pass when a pushbutton is

actuated.

■ Source code and inexpensive development tools allow 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 when purchased as a kit from the NJQRP Club.

"What Can I Do with a PSK31 Audio Beacon?

As 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 the DigiPan application. 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 is customized with a callsign and is run on slightly different frequencies. One Beacon may have its tones centered at 978 Hz, while another may have it's tones at 1050 Hz. This spreading out of Beacon signals will help when multiple beacons are placed in service at the same time within close geographic proximity.

Okay, so you've got the picture of a whole bunch of club members amassed around a table with an operator sitting at the laptop running DigiPan, right? Within a specified period of time, the idea is to see how many of the Beacons can be copied (callsign & code word), as captured by DigiPan. 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), etc. "Points" are awarded to all Beacons for the degree of solid copy captured to the DigiPan log during that 15 minute 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 Forum held last spring.)*

This all may 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 activity, but the PSK31 Audio Beacon has lasting value for the PSK31 enthusiast. The audio tones generated by the Beacon can be presented as input to any SSB transmitter — a Warbler, PSK-20, PSK-40, or even a Yaesu FT-1000MP. Thus, a PSKer can easily put a beacon signal on the air for all the same reasons that CW beacons are used (e.g., 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 needs to be extremely low compared to the 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. One should take the output of the R-2R DAC and put it through a voltage divider pad (or potentiometer) to bring the 0-to-4.5V sine wave signals down to the millivolt range required by an SSB transmitter. The lower the better! If the transmitter is overdriven, all sorts of problems occur with the transmitted RF spectrum – terrible IMD, distortion, and interference to other signals up and down the band. You will quickly learn the wrath of others seeing your callsign and email address transmitted over and over.

Secondly, even in the case of perfect signal quality, very careful attended operation needs to be done when using this Beacon project with an RF transmitter. There should be no interference whatsoever to any other communication on the band, and even while in the CONTINUOUS mode of operation, the Beacon should be stopped periodically to determine if other signals are present.

Never leave the beacon on for continuous, unattended operation! Unattended beacon operation is illegal in the FCC's eyes, and you would not be able to determine if other signals are present in the area of you transmissions.

CIRCUIT DESCRIPTION

Refer to the Schematic at the center of this manual for the following discussion.

DC Power Input

The kit provides a standard battery clip with which the user can connect a 9V battery. Any DC voltage from about 9-12V may be used, as the 3-pin regulator VR1 drops the input voltage down to the required 5V for the microcontroller.

Current consumption of the Beacon circuitry is nominally about 80ma, so the regulator will naturally get a little warm

Note: If an SX-Key programmer is connected to the Beacon (allowing reprogrammability of the microcontroller) a TO220-packaged 1A voltage regulator such as the LM7805 should be used due to the overall higher current demands of the programmer.

Scenix SX28 microcontroller

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

Carrier Generation: "R-2R DAC"

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 15 resistors whose nodes are fed by an 8-bit parallel output port of the microcontroller. The values of the resistors in the network are 10K- and 20K-ohms, 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 lad-

der is a clean sine wave.

Audio Amplifier

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

This amplifier provides significant gain and quite nicely drives a small speaker.

Configuration Jumpers

Five user-installable configuration jumpers (X2-X6) instruct the software to produce one of 24 distinct carrier frequencies that will ultimately be phase-modulated at 31 baud. 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.

Another jumper (X1) instructs the software to set a specific interrupt timing that permits the software UART in the SX chip to communicate over the RS232 serial line with an external computer. This mode is used for custom beacon string entry or character-by-character sending of data.

Carrier Frequency Selection

a) 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.

b) Carrier Offset Jumpers - Three additional jumpers allow user to select one of 8 closely-spaced frequencies around the chosen base carrier

Transmission Mode: Continuous or One-Time

The Beacon may operate in either Continuous transmit mode, or in One-Time transmit mode.

a) Continuous transmit is selected by installing configuration jumper X7, thus instructing the beacon software to automatically restart the beacon transmit sequence (idle and pre-programmed data string),

b) One-Time transmit is selected by removing the configuration jumper from X7. The transmit sequence is initiated by manual actuation of the STARTpushbutton, 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 STARTpushbutton actuation.

Serial Input Mode

Jumper X1 instructs the software to use a fixed-time interrupt loop, thus allowing the UART routine send and receive serial data with the external terminal/computer. It may be convenient to use a SPST toggle switch for this configuration line, as it would only be used when inputting a custom beacon string or when transmitting keyboard data character-by-character. The switch would be open other times during automatic Beacon transmission.

Custom String Entry Mode

The way to initiate entry of a custom text string is by having the X7 (CONTINUOUS) jumper in place and holding down the START pushbutton while applying power to the board. (You also must have the X0 jumper in place to signify use of the serial COM port, as described above). A convenient way to switch the Beacon into this serial-entry mode is to wire the X7 and START lines to a DPDT toggle switch that would ground the respective input lines when this mode is desired.

SOFTWARE DESCRIPTION

Beacon String Construct

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

The SX chip provided in this kit comes pre-programmed wih the information specified in the inital ordering process. This beacon string is "burned" into the program memory and is tranferred to internal RAM memory when power is applied. Whenever the START pushbutton is actuated or the CONTINUOUS jumper put in place, the string in RAM memory is encoded and audibly transmitted.

Entering Custom Beacon Strings

One may alternatively enter a different text string directly to RAM memory over the RS-232C serial COM port provided on the Beacon board. A separate computer would typically be used to send this new text string to the Beacon. The new text is entered character-by-character into RAM data memory, effectively overwriting the pre-programmed text string that was put there initially at power-up time. But since RAM data memory is volitile (i.e., contents are lost when board power is removed), the custom-entered text string is only present while the power is applied. If power is cycled, the custom text string must be applied again to overwrite the pre-programmed string.

The way to initiate the entry of a custom text string is by having the X7 (CONTINUOUS) jumper in place and holding down the START pushbutton when applying power to the board. You also must have the X0 jumper in place to signify use of the serial COM port. A convenient way to switch the Beacon into this serial-entry mode is to wire the X7 and START lines to a DPDT toggle switch that would ground the respective input lines when this mode is desired. The X0 input line could also be put to a SPDT switch to be grounded for use of the serial port. (Two separate switches are recommended for reasons explained in the section concerning Direct Character Transmission mode.)

Interrupt Timing

For this part of the discussion, it will be assumed 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 software 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 3.94us and the Interrupt Service Routine (ISR) counts two of these interrupts and then signals the presence of a 7.88us 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.

Creating the Carrier Sine Wave

Every 4th time the 7.88us window starts (i.e., at the 31.25us 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 31.25us windows have occurred) a single sine wave will have been constructed within a 2ms 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 31ms interval), the PSK bit window is present and the PSK bit processing begins.

Actually, the recurring 31ms 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.

Creating the Phase Reversals

The bits constituting the 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 31.25us 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 7 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 31.25us 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 oscilloscope screen photo in Figure 1 on the next page.

Here we see 15 cycles of the 500 Hz carrier constituting a 31ms bit-processing window. The sequence of characters processed were two sequential zeros – the first being encoded in the phase reversal seen zt 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 7 cycles of the next window.

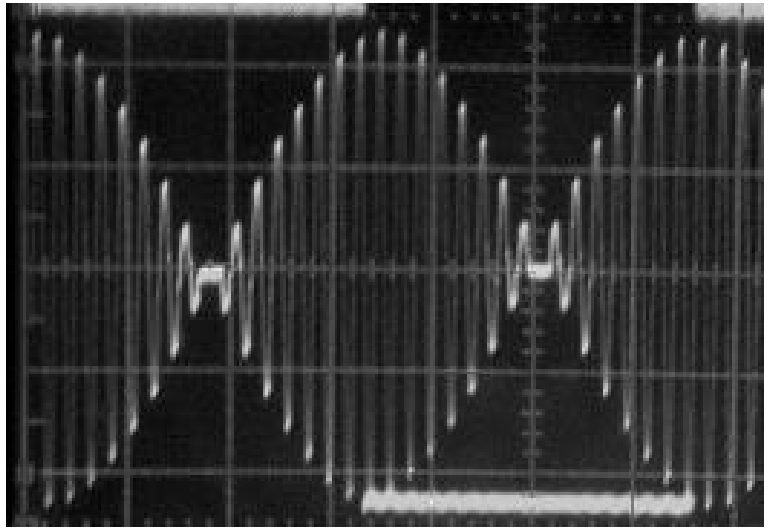


Figure 1: Oscilloscope trace of 31.25 ms timing window (square wave on channel 1) superimposed on the modulated 500 Hz carrier wave on channel 2. Note phase shifts at the zero-crossings, indicating two successive logical 0's in the digital bitstream.

Encoding a "T"

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 sketch in Figure 2 illustrates what one would 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 the diagram below, 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

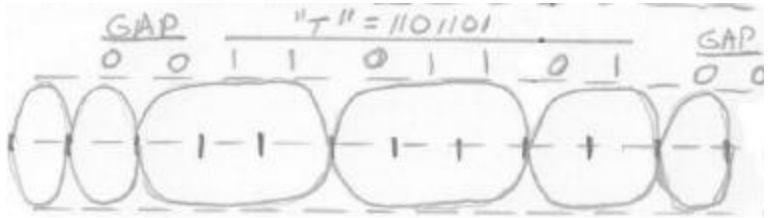


Figure 2: Varicode "T" waveform

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.

One can easily see these principles in action in the oscillosope trace in figure 3. The phase reversals can be seen at the point where the power is reduced to zero, which is at the bit-processing intervals (31ms). The square wave superimposed above and below the carrier is the 31ms sync signal present on the Test Point of the processor (bit 3 of port RA.) It can be clearly seen how the current/next bit processing is actually done at the mid-bit position, halfway through the 31ms period window

Producing Different Frequency Carriers

The preceding discussion assumed a 500 Hz carrier and a core interrupt time of 3.94us. 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 7.88us sine wave construction. Then, based on the state of the configuration jumpers, we're able to sample at half-rates to get the 15.75us rate for 1 KHz carrier generation, and 31.25us rate for 500 Hz carrier generation.

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

The Real Time Clock Counter (RTCC) is clocked by the master 50 MHz oscillator internal to the microntroller. 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, RTCCvariable 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.

BASIC ASSEMBLY

There's really not too much to assembling this project – it's really just a couple of IC's and a handful of resistors and capacitors mounted on a printed circuit card. All you'll need to do is to assemble the circuits according to the Schematic and Parts Layout diagrams, plug the chips into the sockets, connect a DC power source and a speaker, and the Beacon should work. No dreaded toroids to wind, no alignment … no muss, no fuss!

The top of the actual pc board is noted "Component Side". Start by inserting the IC sockets into their respective holes. Carefully solder all pins in pace on the bottom side of the pcb.

Next solder the J2 DB-9 connector in place.Be sure to solder the heavy tabs that will protrude through the two large holes in the pcb, as this will give the connector strength.

Next install the resistors. All resistors will be mounted in the "upright" position, as noted in the sketch on the Layout diagram. Although there is no silkscreen on the pcb to denote precise location of components, carefully matching up the hole patters to the Layout diagram should allow an easy install of the resistors.

Install the 3-terminal voltage regulator, VR1. This component should be mnounted so it can be bent over parallel to the pcb and about 1/8" off the board. Although it will get a little warm during operation, no heatsink will be required.

Install all remaining components: capacitors, the 3-terminal resonator Y1, and the potentiomeent R4.

Carefully wire all off-board components to their respective pads, following the Layout diagram. This includes the speaker, the START pushbutton, the J1 battery clip,

Before inserting the IC's to their sockets, apply power to the circuit and ensure that the "O" (output) pin of VR1 reads 5V DC.

Turn off power and insert the ICs to their respective sockets. Ensure that pin 1 of each IC corresponds to the Layout diagram.

With all off board components connected, you can 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. Adjust R4 for an acceptable volume level.

The "acid test" of your success will be to power up a computer running DigiPan or other PSK31 program and present the Beacon audio as input to the computer. Many computers and most laptops have a built-in speaker 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 waterfall 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 re-check 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 15 Hz square wave (31ms high, 31ms low) that can be seen as a voltage bouncing around between

2-to-3 volts on a DC voltmeter. If you have an oscilloscope, you should see the PSK31 waveforms at the output of the DAC, as indicated on the schematic. You will also see a relatively constant 2.4V reading at this same point when using a DC voltmeter. The PTT pad will be at 5V during transmission and at 0V when the Beacon is stopped. Current consumption of the Beacon is nominally about 80ma.

Using the Serial COM Line for Custom Beacon String Entry

You can next check out the capability of entering a custom beacon string.

The serial channel is set for 19,200 bps, no parity, 8 bits of data, and 1 stop bit ("N-8-1"). Your external computer terminal should be set in this way also.

As described previously, put the X1 jumper in place to signal Serial Mode.

Put the X7 (CONTINUOUS) jumper im place and hold down the START pushbutton while turning on the power to the board.

The terminal program running on the computer will display a ":" (colon) character to prompt you to begin string data entry.

You may enter up to 64 characters from the keyboard, or you may terminate the string with the ENTER key before 64 characters have been entered.

The upon string entry completion, the software will immmediately start transmitting the new beacon string (because the CONTINUOUS jumper is in place). You may remove that jumper and use the START puhsbutton to initiate another string transmit.

You may also set or remove any jumpers to shift the carrier frequency of the Beacon as previously described.

Don't forget that the new beacon string you entered is "volitile" and will be lost upon removal of board power. In this event, the Beacon will revert back to the pre-programmed string.

Using the Serial COM Line for Character-by-Character Transmission

Whenever the Beacon is quiet, it is waiting for an event at the "top" of the program. Normal events are the START button being actuated or the CONTINUOUS jumper X7 being put in place.

Another event that triggers transmission is data coming over the RS232 serial COM line. In this case the character received is immediately encoded and transmitted out to the speaker/MIC.

Buffering of characters is not accomplished in the current version of Beacon software - you must be careful to type slowly so as to not send a keyboard characeter before the current character is completely transmitted. This 31 baud rate will seem slow at first but one is quickly able to adjust the typing speed. Future versions of the Beacon software will provide a buffering of input characters to enable "type-ahead".

Again, as in the case of the Custom Beacon String Entry mode, you must ensure that the computer terminal program is set for 19,200 bps, no parity, 8 bits of data, and 1 stop bit ("N-8-1").

You must also ensure that the Serial Mode jumper X1 is in place.

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.

Stay tuned for a single-chip demodulator design in progress to serve as a companion to this Beacon modulator chip, which will then serve as a complete modem for a standalone PSK31 controller!

REFERENCES

1. Scenix programming tools, manuals, parts and other technical information:

2. Parallax — http://www.parallaxinc.com … tools, eval boards, programmers, parts

3. Ubicom (formerly Scenix) – http://www.ubicom.con

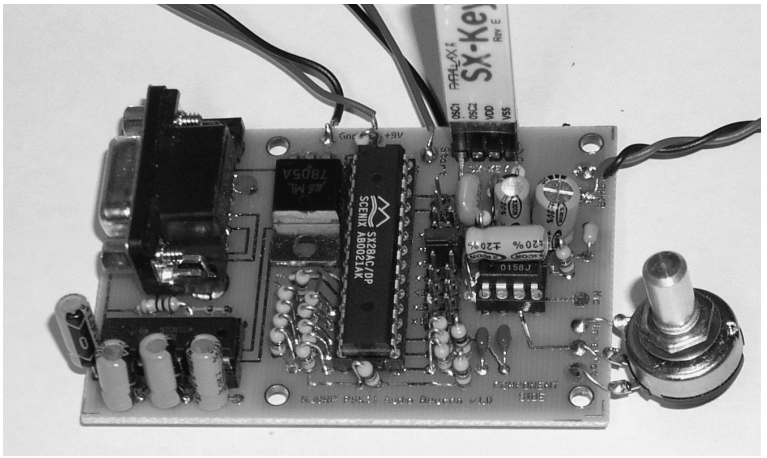
4. SXTECH – http://www.sxtech.com

5. SX Forum – http://www.sx-forum.com

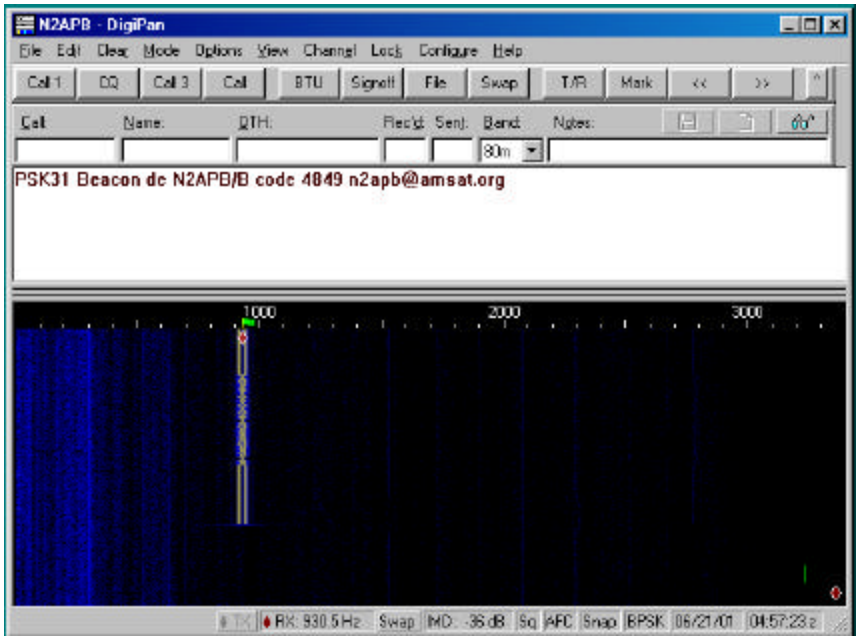
6. Beacon Kit online information website, containing source code, manual revisions, kit notes, tips & techniques, example photos, etc … http://www/njqrp.org/psk31beacon/psk31beacon.html

7. See the Official PSK31 website for a robust listing of PSK31-related articles, technology descriptions, etc. Located at http://www.psk31.html

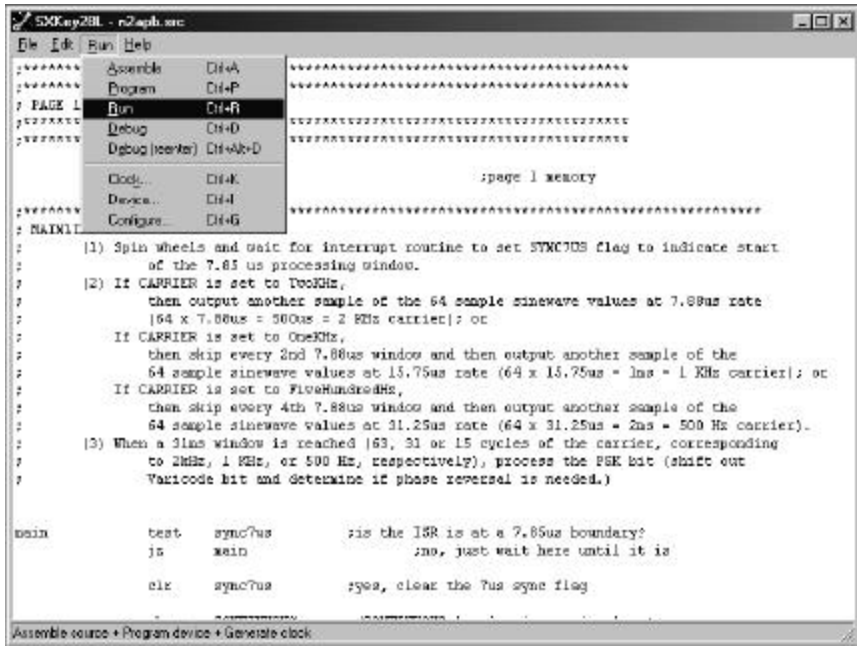
Special thanks are due to our club technical advisors Joe Everhart N2CX and Dave Benson NN1G, for their generous support during the development of this project.



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 PC. 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 is mounted in an unusual manner over the LM386 audio amp on the right - the final layout of the pcb provided in the kit brings this cap down to the board.



This is how the DigiPan screen looks when receiving an audio PSK31 Beacon signal via the computer's sound board (e.g., through the laptop's built-in microphone). Note the characteristic dual-tone "railroad track" signal during the leading zeros sent for synchronization, and its relatively clean indication of -36 dB IMD. DigiPan displays the incoming PSK31 signal in the receive window, showing the pre-programmed data string used in the auhor's prototype.



This is a view of a typical "development environment", simply consisting of an integrated editor/assembler/programmer/debugger from Ubicom, the supplier of the SX chips. Once the program is entered or modified, the user selects the Run menu option as shown which initiates the assembly operation. If there are no errors in the program, an automatic chip programming sequence is initiated with the target board through the SX Blast or SX Key cable, and the program is then run on the target. Overall time for this assemble/program/run operation is about 15 seconds.

APPENDIX A
Programming the SX28 Microcontroller

Developing software programs for the Ubicom/Scenix SX-xx series of microcontrollers is a fun and relatively simple process. The only tool needed is called the "SX Blitz Programmer", which is a small 1" x 1/2" board that fits on the end of a standard DB-9 serial cable coming from a PC. The other end of the SX Blitz plugs into the a 4-pin header on the target project board (i.e., our PSK31 Audio Beacon). The CD-ROM that comes with the SX Blitz contains a combined editor/Assembler/programmer application that is used to "burn" your custom software into the flash memory of the SX device.

To use this simple development environment, just connect the serial cable and SX Blitz combo to your project board, and enter your assembly language source code of your program (or load the supplied source code file audiopsk31v1.src" from the ARRL website). Then select the menu item "Program" which assembles your source code file and programs the target SX chip all in one quick operation. The SX Blitz board sells for $79 from Parallax, and full details can be seen on their SXTECH web page at www.parallaxinc.com/html_files/sxtech/sxtech_home.htm. Other SX tools are also available that provide valuable in-circuit debugging capabilities (using the SX-Key) as well as demo and evaluation boards.

APPENDIX B
Warbler's Guide to Considerate Beaconing
by Joe Everhart, N2CX

Operating a beacon can truly be fun. One can determine antenna performance, propagation characteristics, effects of different power levels, etc. However it's not something done every day and many will never use a beacon on the air. But if you do couple your PSK31 Audio Beacon with an SSB transmitter, here are some guidelines for considerate beacon usage.

Any use of a transmitter to radiate a signal should be done with courtesy in mind. The electromagnetic spectrum is like real estate – it's a limited resource that we share with others. And the radio amateur bands are like limited access parkland to which we are granted access so long as we use them properly. Other licensed hams want to enjoy them too and we must keep that in mind so that we don't lessen their pleasure by our use of the bands. After all, you wouldn't play baseball in the middle of a picnic ground and picnickers are unlikely to spread their lunch out on a ball field!

As mentioned, beacons are not an everyday tool. When they are put on the air it's for a specific purpose — often to gauge signal propagation. This means that we do want them to be heard by someone.

Beacons may simply be an unwelcome intruder to others. Only shorts periods of operation should be limited to only that short period of time when it will do some good without being and annoyance. Similarly, it is extremely impolite to just plop it on the air willy-nilly and let it transmit for extended periods when band activity is high. Its use to judge propagation is questionable when a band is highly populated; simply "reading the mail" on existing signals gives much the same information.

Then, too, at least on the HF bands, running a long duration beacon is not a legal activity. Only on VHF and higher (in the US) can one run a full-time beacon and then only under certain conditions. To remain legal one has to constantly monitor the operating frequency to ensure that no interference is caused by the beacon and operation must cease if such interference is likely to occur. This means that there must be a full-time operator monitoring the frequency for the duration of beacon operation. Continuous or unattended operation is clearly illegal and will likely gain the FCC's attention in addition to arousing ire on the part of other band occupants.

To ensure legality the beacon must use a valid callsign for operation and any transmissions must take place on an appropriate frequency for the class of the operator's license. Abuse of this will be very evident to any government monitoring station or ARRL Official Observer.

Another important consideration in coupling this PSK31 Audio Beacon with an SSB transmitter is the audio drive level feeding the transmitter. The design of this project accommodated a high level audio power amp in order to drive a speaker, however this speaker drive level is far in excess of the proper drive level for a transmitter. You should at least use the direct output of the R-2R DAC as the input to the transmitter, and your will likely even need to pad it further with a potentiometer. The "splash" caused by overdriving an SSB transmitter with these audio tones causes severe side channel energy and interference to others near your frequency. Either use an oscilloscope (looking at the RF signal) to determine if you are over modulating the transmitter, or get on-the-air reports from others while backing off the drive level until an acceptable signal is being transmitted.

While you want your beacon to be receivable, you really should not interfere with normal operations on a given frequency. It is in extremely band form (as some CW beacon operators have found) to run a beacon exactly on a frequency commonly in use for other purposes. This

APPENDIX C

Scenix SX Microcontroller – An Overview

A comparison of Microchip PIC to the Scenix SX microcontroller with some historical details

by Allan Owen, WA3OWT

Many people know of Microchip's PIC16xx series of chips, but not as many are aware of the heart of this project — the Scenix SX microcontroller (MCU). Very similar in internal construction, features and software instruction set, the "SX" offer tremendous speed improvements over the PIC – the SX is about 40 times faster! Because of this speed advantage, the SX is able accomplish real time control and signal processing that begins rivaling the domain of DSP chips.

But first, since so many of us are familiar with the PIC series of MCUs, let's approach the Scenix SX overview in the form of a comparison to the PIC.

Microchip has provided us with a variety of very useful programmable MCU that can be used to design or replace electronic logic and other kinds of control circuits. An MCU is an integrated controller plus memory and I/O peripherals. For many applications in electronic logic and controls the computational requirements are not extensive and the MCU, will do just fine — this is where the small PIC devices really excel.

In fact the "PIC" name that Microchip has given to their parts is an acronym for the words Programmable Interface Controller, a suitable title if ever there was one. These devices are small 8 bit self-contained microcomputers with CPU, program memory, data storage and Input/Output (I/O) functions built in. There were originally three levels of capability and complexity in the line of parts from Microchip, beginning at the low end with the 16 series, the intermediate 17 series and the high end 18 series. These devices had different internal control word lengths and memory array sizes, but in general, the low end 16 series had smaller memory available for both the control program ROM (read only memory) and the internal stored data RAM (random access memory).

A typical early low end device, such as the popular 16C54 had, for example, an Electrically Programmable Read Only Memory (EPROM) size limit of 512 words of 12 bits each. So the control program would be limited to 512 words. The internal RAM for storing data was limited to 25 bytes. The I/O port lines available to connect signals to the outside world, numbered 12 in total. The part was mounted in an 18 pin Dual In-line Package (DIP) or the smaller surface mount equivalent. Higher capacity parts with more resources were offered in larger packages, such as the 16C55 with 20 I/O lines in a 28 pin DIP. Also more program and data memory is available in the 16C56 and '57, with 1K words of program space, 25 bytes of data, and 2K words of program, 72 bytes of data, respectively. Larger memory arrays became available in later series parts.

Even though memory resources are quite limited, a lot of control can be done with these small controllers using the easily mastered instruction set which has only 33 commands. These devices are so powerful and flexible that they can be used to perform logic and control functions that would be literally impractical to do with discrete components and logic arrays. At the same time, the smaller parts can be easily programmed to perform simple control logic that can cost effectively replace the same capability of a handful of TTL or CMOS logic IC's. The parts are made with a modern CMOS semiconductor process which provides the designer with many choices of operating conditions. One being high speed operation, by using a fast clock oscillator for system timing, to obtain fast logic signal response, or conversely, very low power operation, by using a slow clock oscillator, where this option would offer a low current drain, and long battery life in portable applications.

But now we come to some of the important differences between the MCUs of Microchip and Scenix. Most notably, the 50-, 75- and even 100-MHz clock speed capability of the Scenix SX far exceeds that of the Microchip PIC, which will run at most at 20 MHz. (The original "flash" memory version of the PIC, the 16C84 and later revised into a 16F84, can only be run as high as 10 MHz.) Newer versions of PICs, having larger flash memory have become available, and they do run up to 20 MHz, but both the original 'C84 and 'F84 versions are still very popular.

Programming internal memory and loading program can be accomplished using a serial data loading mode supported by a few IC pins, so it is possible for both manufacturers to use a special in-circuit programming circuit design, allowing fast and easy program loading. The PIC program memory of these original parts is EPROM technology and can only be programmed once in the normal molded plastic package (called OTP, for one time programmable). Development parts use a ceramic package with quartz windows for erasing the program memory with UV light energy. As each new version of the software is written, it is assembled or compiled and the written into the program memory to test and evaluate. To change it then, you must erase the old code from memory, by flooding the chip with UV light energy for 20 to 30 minutes, and then electrically writing in the new program, referred to as "burning" the EPROM memory. This software development method has been used with small micro-controllers in embedded systems since the early days of microprocessors about 30 years ago, and it is still facetiously referenced as the "burn and learn" method.

Flash memory of today's MCU's has become very useful in rapid development projects where frequent software changes require lots of reprogramming. The flash parts allow very quick reprogramming, actually writing over the old program using electrical signals, and then electrically reprogramming the data into EPROM memory. All of the Scenix parts are made with easily reprogrammed flash memory, which is electrically erasable or rewriteable, for the program space as well as turning on or off many other chip features such as the oscillator speed, code protection, brown out detection and other items. This is the same EEPROM or flash memory technology available in the Microchip '84 and most later parts.

The Parallax company in Rocklin, California, developed an amazing product called the Basic Stamp using these early PIC MCUs as the control device. In fact they made a small single in-line (SIL) pc

board with voltage regulator, micro-controller with clock oscillator, memory and I/O lines. This self-contained board had everything built and ready to use and at a very attractive price of around $30. It operated from a 9 volt transistor radio battery for hours. You could communicate directly with it by using a special cable connected to your PC and it was programmed to respond to a specialized and limited version of Basic. This popular and easy to use, high level language was stored in a separate non-volatile memory IC, outside the micro-controller on the Basic Stamp. The very successful Stamp product is still in production and has seen many improvements including a big brother version providing more functions, and features such as more I/O lines and memory for the stored program. During the development of these products, Parallax apparently got technical information and architectural details from Microchip about the internal features of the MCU, a factor which undoubtedly helped them develop better products, but also came back to haunt them in the future.

Computer Science teaches that there are two fundamental architectures used in early computers and likewise in many our microprocessors and MCUs today. The first type is the Harvard architecture which uses at least two separate memory arrays for storing the control program information and the internal data that is being processed. This has a potential speed advantage when the memory can be made from fast RAM and ROM, and also if the instruction set that must be decoded from the program memory by the processor can be reduced in complexity. This is called a RISC processor for reduced instruction set computer, and most DSP chips being used today owe their performance to factors such as those described here. The second type of popular computer architecture is Von Neuman which has one combined memory that freely mixes the program commands with the data. In fact everything looks like one big memory array. This computer architecture may appear to be simpler than Harvard type, because the CPU treats everything including I/O in the same way, just like memory, and in fact most mainframes and minicomputers are structured this way, with claims that cost reduction is afforded by this approach. The penalty of course, is that the CPU needs to unscramble the code from the data. So it must do some de-multiplexing while performing those sequential op code fetches. For this reason, the computer probably cannot run as fast in this form and commands are not often simplified, since the system must be able to do many kinds of data manipulation directly into and out of memory and so you now have the complex instruction set computer (CISC type).

The microchip PIC controllers are RISC type and all of have Harvard type architecture, with separate internal program and data memory. These are in fact of different bit lengths or widths, as the data memory is 8 bits wide and the program memory is 12 bits wide in the 16C series of parts. As far as the performance of the PIC micro-controllers is concerned, the early parts could operate with up to 20 MHz clock oscillator, by using either an external crystal or ceramic resonators. The processor internal hardware uses a four phase clocking scheme, and the result is that using the maximum external clock speed of 20 MHz, for example, the internal processor clock is running at one fourth this rate, or 5 MHz. The micro-controller would execute most of the program command instructions in one or two instruction cycles, so the parts had fairly high performance rating of several MIPS, or machine instructions per second.

Then a few years ago, we hear from Scenix in Santa Clara California. They announce availability of a new MCU that is compatible with the Microchip PIC but operates faster, at up to a 50 MHz clock, and even for a time, they advertised a 100 MHz clock speed version of the part. This grabs the interest of circuit and embedded system designers and some pc board designers quit and run off to Mexico. But Parallax likes it and uses it to increase the performance of the Basic Stamp products. The Scenix parts are so much faster and care relatively compatible with the Microchip PIC that Scenix and Parallax are sued by Microchip for collusion and theft of intellectual property. The suit has been settled and Scenix was bought by Ubicom, a company focusing on IC development for internet connectivity of computer equipment and embedded systems. Today the Scenix parts are doing well in production, have had some upgrades, and are well supported in terms tools needed to develop products. Parallax actually makes the "eval", programming and demo boards for most Scenix customers. These are very inexpensive and include good technical documentation for the new user.

Similarities and Differences

The new Scenix micro-controllers are signal-for-signal compatible with the original Microchip PIC MCUs. In some cases, the package is mechanically different, as shown on the data sheets. One package difference to watch out for is the use of the narrow "skinny dip" with 28 pins arranged as a dual row of 14 pins spaced only 0.300 inch apart instead of the more standard 0.600 inch spacing. The use of this package type saves board space, material and cost, and has very little down side. Scenix also makes the MCUs in the standard molded surface mount packages. Other than package type and dimensions, there are almost no differences of significance between the two manufacturers' parts.

The software code is compatible too, as the new Scenix parts will run the same op codes and commands as the corresponding Microchip PIC. The same instruction execution speed is available up to 20 MHz clock, in "compatibility mode". The SX parts will run at the much faster clock oscillator speeds too, or down to the low speed clock, in power saving mode. In addition to the fast clock, the SX can run in Turbo mode, where most op codes execute in one instruction time cycle, speeding up program execution considerably from the same code running in a PIC at the same clock speed, where most instructions take 3 clock cycles. The internal memory space is the same for both parts, with the same size program and data memory arrays being used. There are some differences however, mostly to the good. The program memory is all electrically eraseable or flash type, and can be reprogrammed with software changes or updates, up to thousands of times. Software design, testing and revision efforts are improved in the Scenix parts, due to some built-in debugging hardware placed inside the IC, and this works well with simple and inexpensive external hardware and software debugging tools provided for custom development by the designer.

The in-circuit programming (ISP) feature is accomplished more simply with the Scenix parts than with the PIC, and the two programming methods are not compatible. They are both highly effective in being able to make ISP changes after the micro-controller IC is mounted onto the pc board. With most PIC processors, you can add a 5 pin header to bring signals out directly from the board to the programming device. These include

ground, 5Vdc, MCLR* (master reset line) and two I/O port lines, RB6 and RB7. A 5 wire cable can connect your board to an external programming device. With the Scenix part, adding ISP is also easy to do. Similar to the PIC, you can add a 4 pin header to bring out the signal lines needed to program the parts. These include ground, 5Vdc, and both oscillator pins (Osc1 and Osc2). The external programming device operates through these lines, and again by using a special cable to an external program device, you can change your control program easily after the equipment is built.

Some specific differences between the PIC MCUs and the Scenix SX28 include several new programmable features in the SX28. For example, the clock oscillator has a programmable gain to help use the device with a wide range of external resonators and crystals. Also, internal circuits to provide the I/O port lines with certain electrical attributes, such as triggering interrupts on change of state, pull up resistors and Schmitt inputs to give clean logical signal response to analog inputs are offered. Other hardware differences include the option to use a deeper 8 level stack instead of the two level stack available in the PIC. In software, significant differences exist with the Scenix parts that allow certain software changes to give high performance capabilities in the program that cannot be done easily with a PIC. Ten additional instructions are available in the Scenix parts, allowing easy bank and memory page switching, and interrupts with context save/restore, making possible faster data table look-ups with less code, and direct control code access which has advantages in fail safe software checking and a few other applications.

Today the Scenix family of micro-controllers includes several replacements for the smaller 18/20 and 28 pin Microchip parts. These offer much faster execution times and they are not much more in cost. Both Microchip and Scenix have new higher capacity MCUs with larger memory, more internal features and external I/O pins. These of course, require larger packages offering 40, and 48/52 pins. These new parts and the tools needed to use them are available from Microchip and their authorized electronic distributors. Scenix also provides support tools, development boards and kits, from distributors and the Parallax Company. For example, a complete development including a sample MCU from Scenix is only around $149. A programming and de-bugging tool is $99 and just a programming tool is available for $79 or less.

Both companies a free and highly integrated development software for PIC designers. The Microchip version, called MPLAB, includes an integrated assembler, linker and loader for the programming device. Scenix takes the same approach to helping the designer to use these parts. Scenix provides development software that includes an assembler, loader and programming tools that work well with inexpensive external hardware for programming and debugging all of the micro-controllers in the family.

Visit www.scenix.com, or www.ubicon.com for engineering application notes, data sheets and code listings for popular functions of the Scenix MCUs used by other developers. And make sure to check out the tools, development kits, educational boards and devices at www.parallaxinc.com. Other technical support and information can be found at www.sxtech.com and www.SX-forum.com.