Bob Simmons, WB6EYV

1445 Harbor View Dr, #143, Santa Barbara, CA 93103; **pelican2@silcom.com**

A Two Meter APRS Beacon Transmitter

A flexible transmitter that's ideal for experimenters.

This article describes a 2 W, 2 meter Automatic Packet Reporting System (APRS) beacon transmitter that reports GPS position at regular time intervals. Major parameters (call sign, APRS symbol, and so on) are set via RS232 and saved in nonvolatile EE memory. It requires an external GPS with RS232 output of NMEA GPRMC messages.

This design uses an Analog Devices ADF7012 transmitter chip, driving a Philips PD85004 RF FET, with a PIC 18F442 microcomputer for supervision, packet generation and control. The size is 1.2 × 2.2 inches and weighs just 0.24 ounces. The RF output at 12.6 V dc is 2 W, with a transmit dc load of approximately 400 mA (15 mA between transmissions). The PIC assembly code, CAD files and more are available on the *QEX* files website at **www.arrl.org/qexfiles**.[1] The schematic diagram is shown in Figure 1.

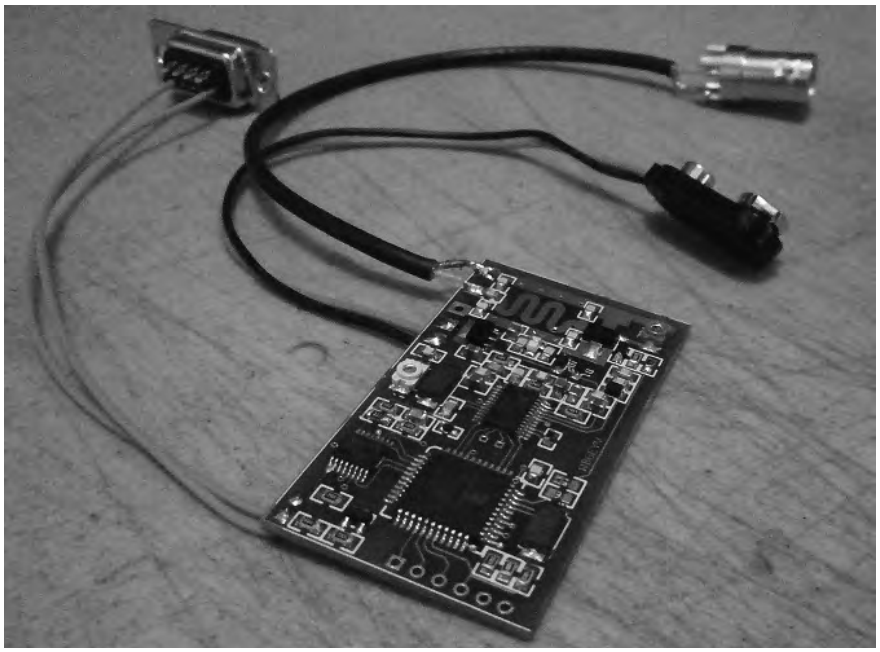

Photo A — Here is the completed beacon transmitter circuit board with programming, power and antenna cables connected.

Construction

Surface mount technology is employed throughout the design. The PIC chip is a TQFP package with a pin spacing of 0.030 inches, and the ADF7012 chip is an SSOP package with a pin spacing of 0.025 inches. Most other components are 0805 packages (0.080 × 0.050 inches), but the varicap diode is pretty tiny. The use of low profile components (including the two crystals) yields a transmitter with an overall profile height of only 0.13 inch.

The method I use to install ICs like these is best described as "glob and wick." Opposite diagonal (corner) pins of a chip are tacked down with solder to hold the chip in the proper position on the circuit board, followed by a close inspection to verify the pins are properly centered on the board footprint. The remaining two diagonal corners are then tacked down with solder to fully immobilize the chip.

A shameless glob of solder is then applied liberally to all the pins (shorting them all together) and allowed to cool. The excess solder is then wicked away using a clean section of solder wick laid across all the pins and heated so that *all* pins are simultaneously "wicked clean" by the capillary action of the wick. If everything is clean, the result is consistent and reliable and not too difficult. The gap between pins should be examined (closely) to verify no solder bridges exist. If bridges are detected, the pins can be "re-globbed" and wicked again.

For Experimenters

This beacon operates on 2 meters, but the basic design is very flexible and (in theory) can be adapted for operation on other bands from 100 to 1000 MHz. Provisions are made for experimenters; all unused pins on the PIC micro are terminated into uncommitted

[1]Notes appear on page 36.

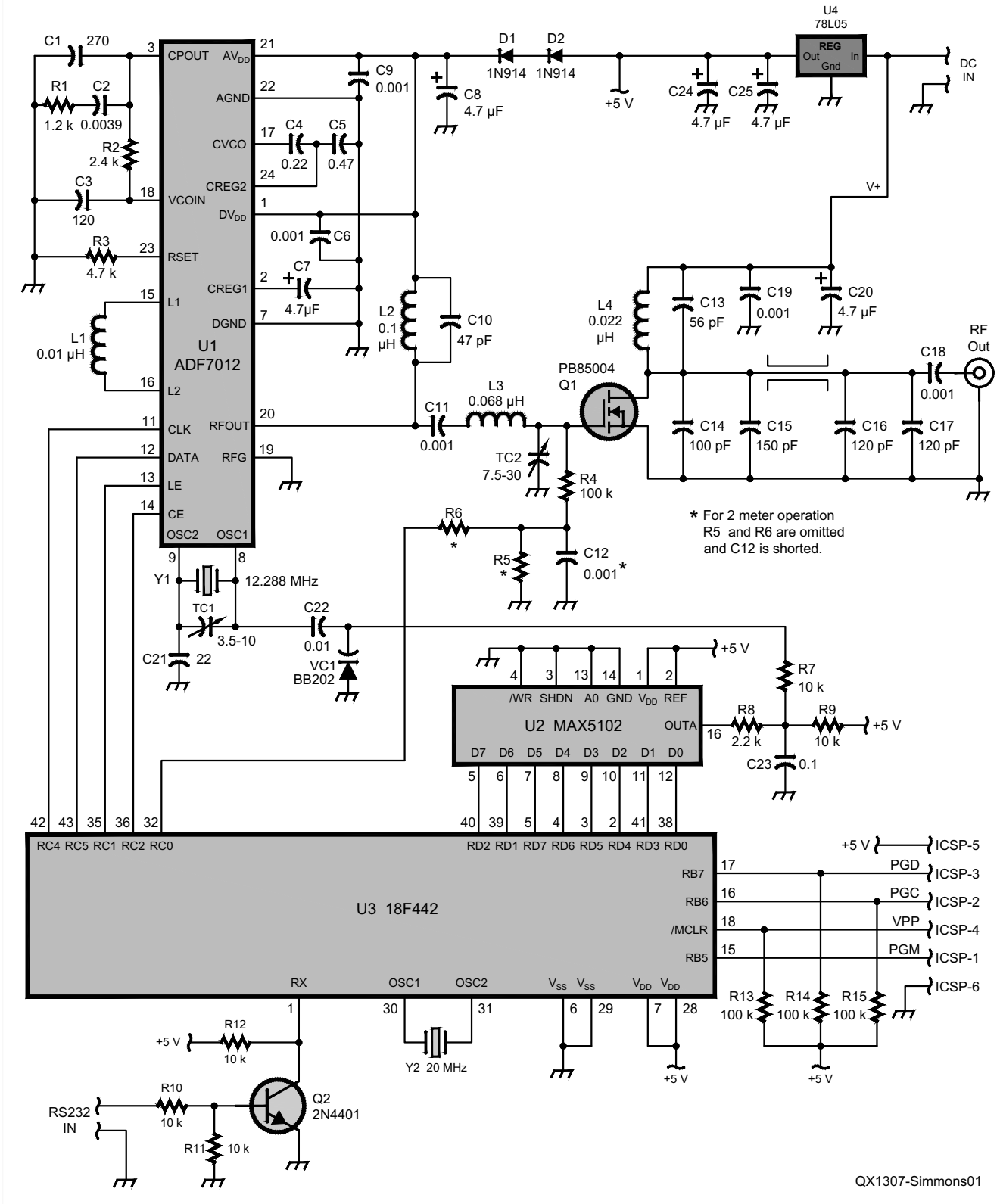

Figure 1 — This schematic diagram shows the beacon transmitter.

Parts List

Quantity	*Designation*	*Description*	*Part Number*	*Cost Each ($)*	*Extended Cost ($)*
1	n/a	Circuit Board (ExpressPCB)	Microbeacon.Board	11.00	11.00
1	U1	ADF7012	ADF7012BRUZ-ND	4.03	4.03
		ALT (Avnet)	ADF7012BRUZ	3.33	
1	U2	MAX5102	MAX5102BEUE+-ND	3.68	3.68
1	U3	18F442	PIC18F442-I/PT-ND	8.74	8.74
1	U4	78L05	296-13531-1-ND	0.38	0.38
1	Q1	PD85004	497-8291-ND	3.51	3.51
1	Q2	MMBT4401	MMBT4401FSCT-ND	0.21	0.21
2	D1, D2	1N4148	1N4148WSFSCT-ND	0.15	0.30
1	VC1	BB202	568-1953-1-ND	0.50	0.50
1	Y1	12.288 CS10	535-9837-1-ND	1.31	1.31
		ALT	XC1660CT-ND	1.69	
1	Y2	20.000 CS10	535-9843-1-ND	1.31	1.31
		ALT	XC1667CT-ND	1.69	
1	R1	1.2 kΩ 0805	P1.20KCCT-ND	0.10	0.10
1	R8	2.2 kΩ 0805	P2.20KCCT-ND	0.10	0.10
1	R2	2.4 kΩ 0805	P2.40KCCT-ND	0.10	0.10
1	R3	4.7 kΩ 0805	P4.70KCCT-ND	0.10	0.10
5	R7, R9, R10, R11, R12	10 kΩ 0805	P10.0KCCT-ND	0.10	0.50
4	R4, R13, R14, R15	100 kΩ 0805	P100KCCT-ND	0.10	0.40
1	C21	22 pF 0805	311-1103-1-ND	0.10	0.10
1	C10	47 pF 0805	311-1107-1-ND	0.10	0.10
1	C13	56 pF 0805	311-1108-1-ND	0.10	0.10
1	C14	100 pF 0805	311-1111-1-ND	0.10	0.10
3	C3, C16, C17	120 pF 0805	311-1112-1-ND	0.10	0.30
1	C15	150 pF 0805	311-1113-1-ND	0.10	0.10
1	C1	270 pF 0805	311-1116-1-ND	0.12	0.12
6	C6, C9, C11, C12, C18, C19	1000 pF 0805	311-1122-1-ND	0.18	1.08
1	C2	3900 pF 0805	311-1132-1-ND	0.10	0.10
1	C22	0.01 μF 0805	311-1136-1-ND	0.10	0.10
1	C23	0.1 μF 0805	311-1140-1-ND	0.10	0.10
1	C4	0.22 μF 0805	587-1287-1-ND	0.12	0.12
1	C5	0.47 μF 0805	587-1290-1-ND	0.12	0.12
5	C7, C8, C20, C24, C25	4.7 μF 0805	587-1782-1-ND	0.24	1.20
1	TC1	3.5-10 pF	490-1996-1-ND	1.20	1.20
1	TC2	7.5-30 pF	490-1994-1-ND	1.20	1.20
1	L1	10 nH 0603	PCD2002CT-ND	0.14	0.14
1	L4	22 nH 0805	495-1850-1-ND	0.79	0.79
1	L3	68 nH 0805	PCD1170CT-ND	0.33	0.33
1	L2	100 nH	PCD1172CT-ND	0.33	0.33
TOTAL COST					**$44.00**

pads, and connections are provided for in-circuit reprogramming of the PIC micro.

Beacon Parameters

Major parameters in the APRS message are user-defined via RS232 by invoking a special "programming" mode of operation. This is accomplished by installing a wire jumper between pins 1 and 6 of the ICSP pads, followed by a power-on reboot. Programming can be done with *HyperTerminal*, but a "utility" program created for this purpose (called *WinBeacon*) is available on the author's website.[2]

The parameters are saved in the PIC nonvolatile EE memory and include these items:

- Transmit frequency (1 kHz steps)
- Transmit repeat time interval (seconds)
- User's call sign and SSID
- Digipeater ("via") call signs and SSIDs (if any, max = 2)
- APRS map symbol (2 characters)
- Comment (32 characters)

All parameters are encoded into a single line of text that is terminated with a carriage return and sent via RS232 to the transmitter. Upon detection of this text, the new parameters are processed and saved in EE memory. The transmitter then switches to the new frequency and generates a "test" transmission for 9 seconds. (An unmodulated signal for 3 seconds, a 1200 Hz modulation tone for 3 seconds and a 2200 Hz modulation tone for 3 seconds.)

Successive programming messages will overwrite the old parameter data and invoke another test transmission, using the new data. To exit the programming mode, the wire jumper is removed and dc power is briefly interrupted to generate another power-on reboot, which causes the transmitter to restart in the regular APRS operating mode (RS232 input = GPS).

Note: It is important to ***not*** connect a GPS receiver to the RS232 input until the programming jumper has been removed and dc power has been interrupted (in that order). Failure to do so will lead to garbled programming data because the first message transmitted by the GPS will be interpreted as a programming message.

The Transmitter Circuit

The transmitter employs an Analog Devices type ADF7012 low-power transmitter IC with about 25 mW output, which drives a Philips type PD85004 RF FET. The FET is operated "approximately" in Class C, to generate 2 W of RF output at 2 meters when powered from 12.6 V dc. The Analog Devices chip employs a fractional-N type PLL circuit, and the PLL loop filter was designed using a free CAD tool available on the Analog Devices website, called *SimPLL*.[3]

Fractional-N PLL circuits are notoriously noisy and the ADF7012 is no exception to this rule. Adjacent channel noise is significant for perhaps 300 kHz on either side of the carrier frequency, (down about 40 dBc), which can cause noise bursts in nearby receivers operating on the same band. The short (and infrequent) APRS transmissions help to mitigate this, but using this design for prolonged transmissions would require attention to this issue.

[Builders should be aware of the FCC Rules requirements for transmitter spurious emissions. In this case Sub Part 97.307 (e) applies:

The mean power of any spurious emission from a station transmitter or external RF power amplifier transmitting on a frequency between 30-225 MHz must be at least 60 dB below the mean power of the fundamental. For a transmitter having a mean power of 25 W or less, the mean power of any spurious emission supplied to the antenna transmission line must not exceed 25 μW and must be at least 40 dB below the mean power of the fundamental emission, but need not be reduced below the power of 10 μW. A transmitter built before April 15, 1977, or first marketed before January 1, 1978, is exempt from this requirement.

As long as the spurious emissions are down 40 dBc as the author indicates, this transmitter would meet the FCC requirements. — *Ed.*]

The ADF7012 chip is programmed and operated with four wires from the PIC 18F442 microcomputer. Four serial messages, each 32 bits long, are sent by the PIC at the start of each transmission. The 12.288 MHz reference crystal is divided by three to yield a phase detector frequency of 4.096 MHz, which allows the microcomputer to easily tune the ADF7012 in steps of exactly 1 kHz. The VCO operates at 4 times the output frequency and is divided down (inside the chip) prior to application to the FET input.

The ADF7012 chip has digital inputs for baseband data modulation of the carrier frequency, but those inputs are not employed in this design. Instead, an 8-bit parallel DAC is employed to generate a modulation sine wave that is applied to the reference crystal with a varicap diode. This is done because 2 meter APRS specifies Bell 202 modulation, which requires audio sine wave modulation of the carrier. The baseband modulation inputs of the ADF7012 chip are strictly digital and do not support sine wave modulation. The modulation characteristics of the varicap diode are nonlinear and the varicap bias resistors have been adjusted (experimentally, with some effort) to achieve the best "balance" between waveform quality, modulation level, and equality of (transmitted) tone levels.

Power for the ADF7012 chip passes through two diodes, to reduce the 5 V supply down to about 3.8 V. While developing this design, my attempts to operate these chips at 5 V yielded several chip failures (data sheet says max = 3.9 V). A 3.6 V regulator was then employed, but I found that the in-circuit programming feature of the PIC chip required at least 4.2 V to work properly. Therefore, the present design uses a 5 V regulator for the benefit of the PIC chip with two diodes in the ADF7012 supply line, for the benefit of the ADF7012 chip.

Provisions are made for applying positive dc bias to the PA stage FET gate during transmission periods (bias = zero at other times). The values of resistors R5 and R6 set the dc bias applied to the FET during transmissions. For 2 meter operation, no dc bias is required, (these resistors are omitted and C12 is shorted), but operation on other bands might require some dc bias to increase the FET gain, by driving it closer to a class B (or even class A) operating point. A trimmer cap in the FET input circuit allows tuning for maximum output power.

The output circuit consists of a Pi network circuit using a circuit board "printed" inductor and fixed capacitors. Output power is typically 2 W when dc power is 12.6 V, but operation at lower voltages is possible. The output power closely follows a square law relationship to dc supply voltage, so cutting the dc supply in half will reduce RF power to about 0.5 W. The voltage regulator for the PIC chip is a garden variety 78L05 that suffers "dropout" at an input of 7 V, so operation below 7 V is not recommended.

NMEA RS232 GPS data is fed to a simple transistor inverter circuit, and then to the UART input of the PIC microcomputer. This input is also used to program the APRS parameters when the PROGRAMMING mode is invoked. The 78L05 regulator uses a 100 mA TO-89 package, with abundant heat sinking to the circuit board. There is sufficient surplus current capacity to provide 5 V dc (if desired) to most GPS "puck" receivers that typically would be used with this transmitter.

The PIC Software

The software is written in PIC assembly code for a PIC type 18F442 microprocessor in a TQFP package, running with a 20 MHz crystal clock. It contains 4000 lines of source code and assembles into 3400 bytes of machine code.

TMR0 is used to generate the Bell 202 modulation tones (1200/2200 Hz). Each sine wave is generated in 63 time steps (one

step per TMR0 interrupt) using a sine wave lookup table and an external 8 bit DAC. The time interval between successive steps is controlled by the TMR0 interval, which can take one of two different values: one for a 1200 Hz tone, another for a 2200 Hz tone. The reload value for TMR0 is fetched from a RAM location each time TMR0 generates an interrupt, so changing the value stored in this RAM location provides a way to generate a phase contiguous shift between tones.

TMR1 is used to generate the 1200 baud data rate clock for the APRS message bits. The TMR0 reload value (stored in RAM) is modified in the TMR1 interrupt routine, if the next data bit to be sent is a logic "0" bit (logic "0" bits require a tone shift, logic "1" bits do not).

TMR3 is used for the transmit repeat time interval (the time between successive beacon transmissions). TMR3 is (briefly) disabled during transmissions because it (somehow) causes significant interference with the transmitted tones.

The UART (receive only) is used to accept RS232 GPS messages during normal operation, and to accept "parameter programming" messages when operating in PROGRAMMING mode.

Events generated by TMR0, TMR1, TMR3 and the UART RX are all handled in the interrupt routine. The starting address for the interrupt routine is dictated by the internal architecture of the PIC chip, and is (for these chips) equal to 0x08. The CSTART routine (ColdStart, at address 0x0) merely jumps to the main startup routines, to bypass the interrupt routines, which must begin at address 0x08.

The MSTART routine (MainStart) contained the main startup routines that configure the various internal features of the chip, initializes the variables and enables the interrupts. The presence/absence of the PROGRAMMING jumper is also detected in this routine, and if detected, control passes to the HOST_START routine (for changing the APRS parameters via RS232).

The EXEC routine is the top-level routine that supervises operation after completion of the MSTART code. The first part decides if a transmission is required, either a timed transmission or a "forced" transmission, caused by a switch closure to ground on PORTB, bit 6. If a transmission is required, control passes to the EXEC_SEND routine, which then gets fresh data from a GPS message and then calls the SEND_MESSAGE routine to actually generate and send the message.

The GPS message is received and parsed into individual data fields as it arrives, one character at a time, in the UART interrupt routine. The heart of the GPS parsing routine is the RMC_PARSE jump table, which acts much like a SELECT CASE statement found in higher languages. The jump table has 23 possible exit paths, (goto statements) selected by an index variable called GPRMC_STATUS.

As each character (in the GPS message) is detected and parsed in those 23 routines, the GPRMC_STATUS pointer is incremented (when each parsing task is finished) so that the next GPS character that arrives will be "handed off" to the next routine in the parsing sequence. When (and if) the first 22 parsing routines are successfully completed, the last parsing routine (GPS_LF) searches for an ASCII <lf> (line feed) character, which signals the end of the GPS message. This triggers a few more housekeeping chores to adjust the GPS data (if required) and a flag bit is set to indicate that a complete set of valid GPS data is available. This flag bit is tested in the EXEC routine, which waits until it is set before starting an APRS transmission.

Various tests are performed in the parsing routines, to ensure the data is correct. In some cases, the parsing routines will wait for a specific character, and then increment the jump pointer when the character is detected. In other cases, any errors in the GPS data will trigger a restart of the search sequence and the GPRMC_STATUS pointer will be reset to the beginning of the search. Using this method, only a valid GPRMC message will successfully navigate through all 23 parsing routines and cause the flag bit to be set.

A similar jump table scheme is used to generate the APRS message (from a lookup table of characters) in the TEXT_TAB routine located in the SEND_MESSAGE routine.

The actual bytes and bits to be transmitted (once they are fetched) are processed in the SEND_BYTE, SEND BIT and FLIP_TONE routines. The SEND_BYTE routine does the bit stuffing (if required) and calls the CRC_CALC routine for each data bit, to keep the CRC checksum value honest. The TX_START and PLL_SEND routines start up the ADF7012 PLL chip each time a transmission is performed, by sending 4 bytes to each of 4 registers in the ADF7012 chip (16 bytes total).

The HOST routines are only used if the PROGRAM MODE is selected (ground strap was detected when dc power is turned on). The PROGRAM mode is used to set the various parameters of the transmitter, including call sign, repeat time, APRS map symbol, comment and so on. The HOST routines wait for a complete programming message to arrive via RS232, then they parse the message and save the various parameters in nonvolatile EE memory. This is followed by a test transmission.

In-Circuit PIC Programming

For experimenters, provisions are made for reprogramming the PIC chip *in situ* using a simple cable that can be made to mate with a PIC programmer. Six holes (circuit board vias) are located along the top edge of the circuit board near the PIC microprocessor. These holes are spaced on centers of 0.1 inch and are large enough to allow insertion of the 0.025 inch square wire wrap pins, like those typically used on wire wrap IC sockets or garden variety SIP header strips.

In a sense, these holes provide the in-circuit programming socket and the mating strip of wire wrap pins (inserted into these holes) would be the programming plug. A simple cable can be contrived to connect this plug to a PIC programmer to allow in-circuit reprogramming of the PIC chip. (*Hint:* Hold your finger on the plug to maintain good contact between holes and pins while programming the PIC chip. It is a loose, unreliable fit otherwise.)

Five wires are required, but the PIC chip allows two different programming methods (high voltage or "HV" method as well as a "5 V" programming method) so 6 pins are provided to allow user selection of the desired approach. Pin 1 is signified with a square pad; the others are round. The pin definitions are:

- Pin 1 PGM pin (used ONLY for 5 V programming mode)
- Pin 2 PGC pin (programming clock)
- Pin 3 PGD pin (programming data)
- Pin 4 V_{PP} pin (programming pulse, used for HV programming mode)
- Pin 5 V_{DD} pin (+ 5 V dc power)
- Pin 6 GND pin (ground)

For example, using a PicKit 3 programmer, the following cable would work:

PicKit 3	*MicroBeacon*	*Signal Name*
Pin 1	Pin 4	V_{PP} (HV programming pulse)
Pin 2	Pin 5	V_{DD} (5 V supply
Pin 3	Pin 6	GND (ground)
Pin 4	Pin 3	PGD (programming data)
Pin 5	Pin 2	PGC (programming clock)
Pin 6	Pin 1	PGM (5 V programming select)

A PicStart Plus programmer expects an actual IC to be inserted into its ZIF socket, but a cable can be made for external programming by using a wire wrap IC socket as a "dummy" chip, which is inserted into the programmer's ZIF socket. Cable wires running to the beacon programming "plug" can then be soldered to the WW pins of this socket.

Note: To program the PIC chip, the beacon dc power must be turned on. Do not rely on the programmer to provide 5 V dc power. For regular "HV" programming mode, the only connections actually required are V_{PP}, ground, PGC and PGD.

The Circuit Board CAD File

The circuit board CAD file uses the ExpressPCB "MiniBoard" service, and was created with the (free) CAD tools available from ExpressPCB.[4] Their "MiniBoard" service is highly standardized and inflexible, to allow maximum economy, and the CAD file provided here will generate nine beacon PC boards, for a total (bare board) cost (June 2011) of US $65.

The boards are paneled in this design (to comply with MiniBoard requirements) and must be separated from each other using a hack saw, coping saw, or some similar means.

The Parts List

Except where indicated, all part numbers are DigiKey; prices and quantities are for a single beacon transmitter. Parts marked ALT are alternative parts (in case the primary part is not available). Quantities and costs are per transmitter.

Bob Simmons, WB6EYV, was first licensed as a Novice in 1964 at age 13 and remained licensed (more or less) constantly ever since. He also earned a commercial FCC license in 1967. Bob served Naval Reserve duty as a radar technician with about 6 months of total sea time. He spent several years doing civilian work in nautical and marine electronics in Los Angeles harbor, as well as doing some land mobile radio work. This was followed by five years in flight line avionics, working with business jets. He moved to Santa Barbara, California in 1992 and worked on vacuum deposition systems for five years and held assorted engineering jobs at other times. Presently, Bob is self employed and runs a website, making and selling radio direction finding equipment and modules, with the majority of his latest work spent creating embedded software/hardware and developing technologies to enable Internet-linked remote DF stations. His primary interest is developing and applying new technologies to old problems and pushing the DF art forward.

Notes

[1]The ExpressPCB circuit board files and the program code files for this article are available for download from the ARRL *QEX* files website. Go to **www.arrl.org/qexfiles** and look for the file **7x13_Simmons.zip**.

[2]You can download the *WinBeacon* program at: **www.silcom.com/~pelican2/PicoDopp/XDOPP.htm#WBCN**.

[3]The *SimPLL* CAD program is available for download from: **https://form.analog.com/Form_Pages/RFComms/ADISimPll.aspx**.

[4]Learn more about the *ExpressPCB* CAD program and download the files at: **www.expresspcb.com**.