Julieboard

An easy-to-build DDS synthesizer for the PC printer port.

by Bruce Hodgkinson VE3JIL

Every so often, a technology development comes along which radically and permanently alters the landscape of amateur radio—spark to CW, AM to SSB, vacuum tubes to solid-state, and so forth. Each of these new developments has made possible things which could only be dreamed of before, but quickly become taken for granted. The introduction of direct digital synthesis (DDS), a DSP-related technique, has been one such advance in the RF design field. At first, DDS-based gear could be afforded only by the military, but the state of the art has now advanced to the point where new commercial and amateur radio designs include it as a standard feature.

Why DDS?

The best way to answer that question would be to take a look at the disadvantages inherent in the old techniques. Traditionally, VFOs (including those based on phase-locked loops) have employed analog LC oscillators dependent on mechanical and physical characteristics for frequency control. Although analog oscillators are appealing due to their apparent simplicity, they fall prey to the usual analog type bugaboos: calibration error, drift, phase noise, excessive lockup time, etc. This means that designs which use analog frequency control can—and usually do—lead to alignment, debugging and calibration hassles which then require expensive equipment and time to fix. For those working with phase-lock loops, there is the additional problem of making the frequency resolution vs. lockup time vs. capture/lock range tradeoffs, which invariably compromise performance and/or force the designer to go to multiple loops, mixers, filters, etc.

With DDS, on the other hand, a few chips on a board slightly larger than a business card can implement a wideband oscillator which gives:

0 to 16 MHz coverage
0.007 Hz frequency resolution
Virtually instantaneous switching time
No drift/no calibration
Excellent spectral quality
Simple interface via PC printer port

So, I designed one. This board (which I've named the "Julieboard") is easy to build and is intended for use as a building block to add digital tuning capability to home-brew equipment.

Why the PC Parallel Port?

The first part of this question really asks, "Why the PC?" The DOS computer has made its way to a very high number of amateur stations, doing such jobs as log-keeping, packet radio, word processing, satellite tracking, et cetera. The price has come down to the point where (occasionally) first-generation PCs are actually available free for the taking if one is at the right place at the right time! The software required to drive this oscillator is so simple that it will run on any DOS machine, right down to the humblest one-floppy system, which means that if a PC actually has to be acquired for the specific purpose of running this oscillator, it needn't cost more than a nominal amount.

The second reason for using the PC is that it is a fine platform from which to develop and implement control functions via software: The "front panel" can take any form the user wants, changes can be made at will without having to modify or junk hardware, and functions can be easily done which would be difficult, if not impossible, to do with dedicated hardware. Rather than being stuck with one hard-wired approach, the user has a software "playground" in which his only limitations are imagination and time available for programming.

Finally, code can be written, modified, and debugged on the same machine on which it runs—allowing the use of widely available and reasonably priced development tools.

The second part of this question is, "Why the parallel port?" Why not do a plug-in (slot resident) version? The first answer is that not all PCs (lap-tops, for example) have plug-in slots available for another board. Also, many PC owners, especially those without a technical background, are not really keen on tearing apart a working system just to install another board which then has to be configured and set up on a particular address location. This is a real problem if the PC belongs to somebody else or the oscillator has to be moved often.

The parallel printer port offers a "plug and play" alternative: Almost every PC has a printer port and few indeed are those computer users who aren't capable of guiding a DB-25 connector onto the end of a cable. Also, a plug-in board approach forces the user to install the oscillator inside the PC itself—which can cause noise problems, as well as impose limitations on where the equipment can go. With the parallel port approach, the equipment can be located a long distance away from the PC and driven via a long extension/ribbon cable for remote operation. Finally, the parallel port, being non-bus-specific, can be replicated with any simple TTL six-bit register. For example, there is no reason why an appropriately programmed single chip microcomputer (such as a Motorola '68705 or Intel '8051) couldn't replace the PC for those who really object to having to drag around a large, bulky PC just to drive a tiny little board. With a single chip microcomputer, an entire HF rig could be made to fit into a shirt-pocket-sized package!

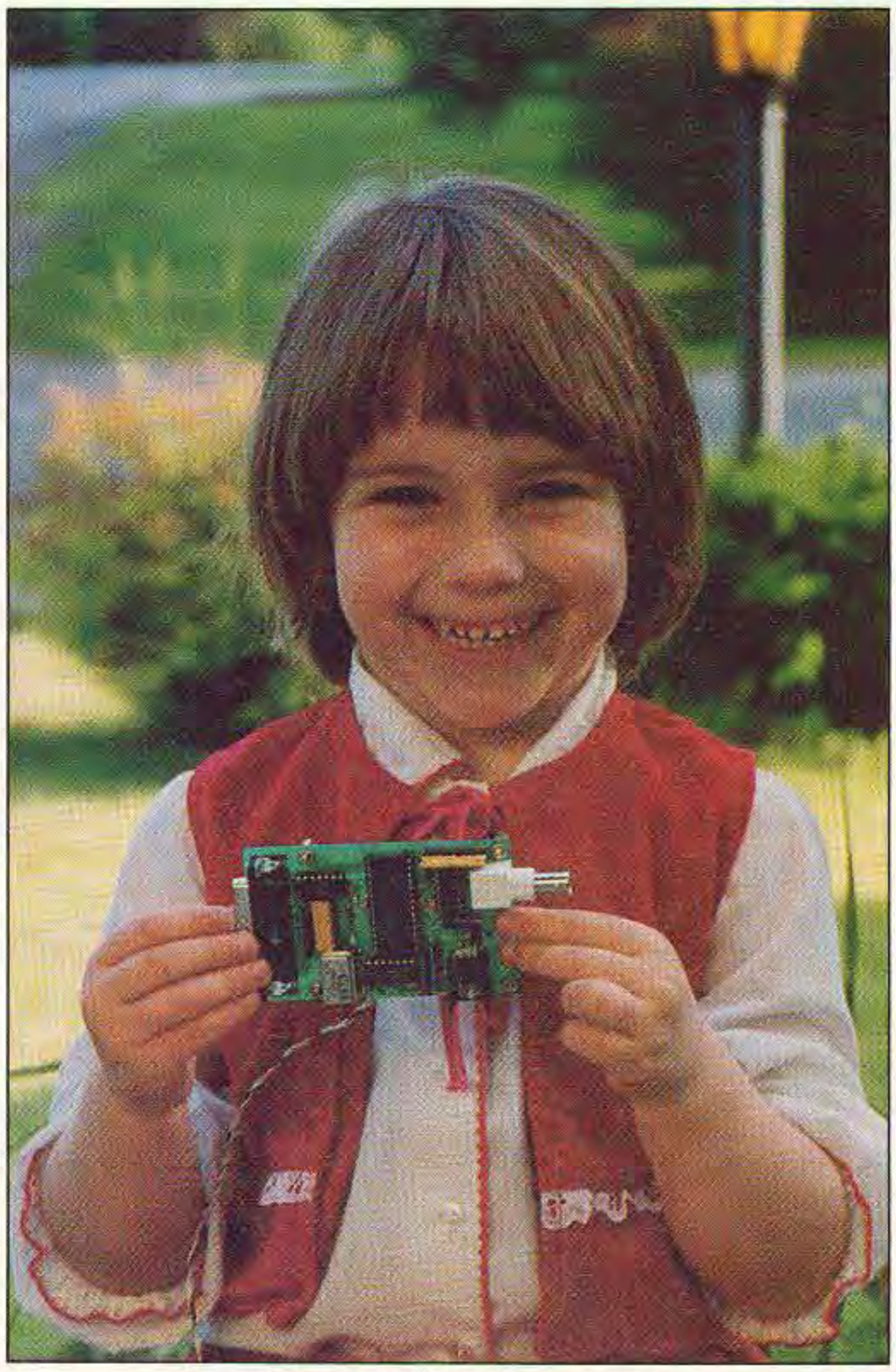


Photo A. Julie and her board.

Circuit Description

The circuitry for the Julieboard fits on a small two-layer printed circuit board about 2.5" x 4.5". On one end is the DB-25 connector for the printer cable and the other end has the BNC output and power/external-clock connectors. Power input needs are not critical—anywhere from about +7 VDC to +12 VDC will do. The input is polarity-protected so if the polar-

ity is wrong, no damage will be done—it just won't work. The incoming DC voltage is regulated down to the +5V level required to run the on-board logic. The only restriction regarding input voltage is to keep it high enough to overcome voltage regulator dropout and low enough to keep regulator power dissipation at a reasonable level. (This circuit draws about 200 mA and the difference between the input voltage and +5V output is dumped as heat at about 200 mW per excess volt).

The largest IC, a 28-pin DIP package (a Harris HSP45102) contains the actual DDS synthesizer logic right up to the sine PROM-output. This device is clocked at a 40 MHz rate by a clock oscillator module and produces a new 12-bit binary word at its output pins every clock cycle. The frequency increment is determined by a pair of internal 32-bit shift registers which are loaded via TTL bit-sequences driven from the parallel port.

The 74HC14 is used as a buffer between the "outside world" and the Harris DDS chip—it provides input signal conditioning and serves as a buffer for the more expensive DDS device. Likewise, the 74F132 performs a conditioning and buffering function between on-board logic and the outside world. It performs an automatic line select function for the external clock: If an external clock signal is applied, the board logic automatically selects that signal, saving the user from having to configure any jumpers.

The output of the Harris DDS chip represents a 12-bit binary sample of the desired waveform at the time of each clock tick; before it can be of much use, this binary output must be converted into an analog voltage. The Harris CA3338 video digital-to-analog converter (a 16-pin DIP package) converts the digital outputs into corresponding analog levels at the 40 million samples per second rate. This level of performance was unheard of several years ago and was one of the reasons why DDS systems were so expensive when they first came out. Things have changed.

The output from the DAC looks like a sine wave made up of little tiny "steps"—256 different levels, to be exact. (One small compromise in this design was made by using an eight-bit DAC rather than a 12-bit DAC, but the four "wasted" binary outputs have such a small impact on the output that the savings in cost easily justified the change. With the 12-bit DAC, the sine wave would be made up of 4,096 different levels of steps.)

"Wait a minute," one might say, "That's noise—I don't want THAT on my transmitter output." Without getting too deeply into sampling theory, let me say that "that noise," is almost completely insignificant. Look at the "made-out-of-little-steps" sine wave again. Think of this as an absolutely perfect sine wave with a superimposed noise consisting of those steps. See how small and how much higher in frequency (than the sine wave) is that noise waveform? It is no problem to filter the noise out—done by the low-pass filter module located on the board.

The filter module implements a seventh order elliptic low-pass filter in a 10-pin SIP package. The space taken by a discrete version of this filter could easily take up half again as much room as the remainder of the circuitry. Since this design is a "building block," why not make it as small as possible?

```
10 CLS:LOCATE 1,1:PRINT                          ' Bruce Hodgkinson  VE3JIL
20 PRINT"  JulieBoard 200 Controller:  JUL200.BAS   Apr. 19/93
30 PRINT
40 PRINT"  Options are:       'F' for new frequency
50 PRINT"                     'Q' to quit"
60 PRINT"                     '<' to increment by 100Hz
70 PRINT"                     '>' to decrement by 100Hz
80 PRINT
90 PRINT"                    Frequency (KHz) = "
100 GOSUB 220
110 A$ = INKEY$:IF LEN(A$) = 0 THEN 110
120 IF ASC(A$) = 46 THEN GOSUB 310          'decrement for "." key
130 IF ASC(A$) = 62 THEN GOSUB 310          'decrement for ">" key
140 IF ASC(A$) = 44 THEN GOSUB 330          'increment for "," key
150 IF ASC(A$) = 60 THEN GOSUB 330          'increment for "<" key
160 IF ASC(A$) = 102 THEN GOSUB 220         'new F for "f" key
170 IF ASC(A$) = 70 THEN GOSUB 220          'new F for "F" key
180 IF ASC(A$) = 113 THEN 500               'quit if q
190 IF ASC(A$) = 81 THEN 500                'quit if Q
200 GOTO 110
210 '
220 LOCATE 15,1:PRINT "New Frequency (KHz)"          ' new F
230 LOCATE 15,21:INPUT NF
240 LOCATE 15,1:PRINT "                                "
250 IF NF >16000 THEN 220
260 IF NF <0 THEN 220
270 LOCATE 9,39:PRINT "                "
280 LOCATE 9,39:PRINT NF:N=NF*1000
290 GOSUB 370:RETURN
300 '
310 N = N - 100:IF N<0 THEN N = 0                    ' decr 100Hz
320 GOTO 340
330 N = N + 100:IF N>16000000# THEN N = 16000000#    ' incr 100Hz
340 LOCATE 9,39:PRINT "                "
350 LOCATE 9,39:PRINT N/1000:GOSUB 370:RETURN
360 '
370 'NX = INT(N* 134.217744#): OUT 888,127           phase incr 32MHz
380 NX = INT(N* 107.374195#): OUT 888,127            ' phase incr 40MHz
390 FOR K = 31 TO 0 STEP -1
400 KX = INT(NX/(2^K)):NX = NX - (KX * 2^K)          ' bit by bit
410 IF KX = 1 THEN 430
420 GOSUB 490:GOTO 440
430 GOSUB 480                                        ' shift bit into DDS
440 NEXT K
450 FOR K = 1 TO 32:OUT 888,223:OUT 888,207:NEXT
460 OUT 888,127:RETURN
470 '
480 OUT 888,222:OUT 888,206:RETURN                   ' shift "1" into DDS
490 OUT 888,223:OUT 888,207:RETURN                   ' shift "0" into DDS
500 END
```

Figure 1. Simple controller routine written in GWBASIC.

Driver Software

One of the nice things about using the PC to drive this unit is that a wealth of software development tools are available. The first thing that comes to mind probably would be GWBASIC or some other BASIC interpreter. GWBASIC was used to get the proto-type up and running and a listing of a simple controller routine is shown in Figure 1.

To operate the board, the driver program must drive six DDS control lines:

SDATA*	(shift data)
SCLOCK*	(shift clock)
XFER	(new value transfer)
ENPHACC	(enable phase accumulator)
SHIFTEN	(shift enable)
BANKSEL*	(BANK select)

In normal operation, the software holds all four lines HIGH—enabling ENPHACC (allowing the oscillator to run) and disabling/idling the other three (SDATA*, SCLK*, and SHIFTEN). The choice of "HIGH" as the normal state is no accident: This allows the output frequency to be set from the computer, then disconnected from the PC without losing the programming. This means that the printer port does not have to be tied up permanently—it can still be used to drive the printer while the synthesizer is still in operation by means of a printer switch!

To load a new frequency, the program disables XFER by driving it LOW, enables SHIFTEN by driving it LOW, then shifts in new data by clocking in 64 bits of updated frequency information via the SDATA* and SCLK* lines. Each data bit must be inverted (SDATA* = "0" to shift a "1" into the DDS) and is clocked-in with each HIGH-to-LOW transition on the SCLK* line. The new frequency pair is transferred into the DDS once software re-enables the XFER line by driving it HIGH again. (If this line is allowed to stay active throughout the shift process, there would be 31 periods during which the output frequency would be set to a bogus value, possibly causing interference far off-band).

If desired, the oscillator can be disabled by setting ENPHACC LOW, though this is not critical.

Selection between the two banks is done via the printer port STB* line. This is an open-collector line, so it can be driven from external equipment or from the PC. The PC can read the status of this line, so it can respond to external events via software. For example, a pair of frequencies could be programmed: one frequency

Continued on page 44

Julieboard

Continued from page 42

for "mark," the other one for "space," then keyed to send RTTY.

In a transceiver VFO application, RIT or split frequency operation can be implemented by loading the appropriate transmit and receive frequencies into their respective banks. (This is a good example of software being used to replace hardware.)

If an external line drives this input, it should be a TTL open collector driver so that it does not cause or suffer damage if the computer should drive the line LOW. This "wired-OR" scheme (where if one source or the other or both drive LOW, the line gets dragged LOW) implements an internal drive/external drive scheme which requires no hardware configuration or setup.

Constructing the Hardware

The circuit is not difficult to duplicate and can easily be built with wire-wrap. Because high frequencies are involved, it must be built with the proper techniques or it will not work at all! If you are not familiar with high-speed logic, a commercially fabricated blank PCB or a wired-and-tested board (available from the author) is probably the safest approach.

A high quality circuit board with a low-inductance ground is an absolute must—my standard technique is to use prototype boards with the fat copper strips running up the IC center lines and bridge the strips crossways with a cross-grid built up out of solder-saturated SOLDER-WICK laid along the board. Don't even think of using one of those copperless "protoboards." Plan the layout ahead of time to leave room for the IC sockets and decoupling capacitors.

The IC sockets must be high quality machined gold contact types—the cheap leaf types are not suitable due their high profile (needless lead inductance) and poor reliability. I have often seen them fail, but I've never seen a gold machined contact fail yet. They are expensive (often costing more than the chips they hold), but they are cheap aggravation insurance.

Place the sockets in their final resting places and wire in their ground pins. These must be near zero in length and the widest practical width. My usual practice is to dedicate the bottom-side copper strip (running up the center of the IC pin-rows) to ground and solder the IC ground pins directly to that.

After all, the IC socket's ground pins have been hooked up and the decoupling capacitors have been installed. Decoupling capacitors MUST be placed at the power supply pins of the HSP45102 DDS chip (8,22), the CA3338 video-DAC (13,16), the clock module (14), and the 74F132 (14). The 74HC14 is not a high-speed part, but it should be decoupled also. Keep the leads of the decoupling caps short—the body of the capacitor should just about touch the power pin being decoupled! An eighth-inch lead length is too long. On the DDS chip, the IC designers conveniently placed a ground pin immediately adjacent to each power pin so that the decoupling caps can exactly bridge power/ground with zero lead length—this dictates the use of capacitors with 0.1" lead spacing. Use 0.1 µF as specified—don't try to "improve" the decoupling by using higher values of capacitance: Higher capacitance values tend be more inductive and have a lower self-resonant frequency. Above its self-resonant frequency, a capacitor looks inductive and could make the situation worse than if it weren't there! Once the decoupling caps are installed, wire up the +5V bus to the sockets. At this point there should be virtually infinite DC resistance between the +5V and ground lines.

Next, install the 7805 regulator and its diodes. A heat sink with thermal compound on the regulator is a must, as it will dissipate about 1-1/2 watts with +12V input and can get hot to the touch. If you know for sure that the input voltage will always be +12V, a 22 ohm 2W series resistor can be placed in the power input line to help drop the voltage and decrease regulator dissipation. My usual rule of thumb: If I can't hold my finger on a heat-dissipating device, it's running too hot.

When building something, it is wise to take a "divide-and-conquer" approach by doing the project stage-by-stage and testing it after each round of construction. This is a good time to make the first test—better to fry one cheap regulator now than a board filled with expensive parts later! Apply power to the unpopulated board and confirm that the regulator output equals +5V and that +5V appears at all power supply pins and 0V shows up at all ground pins. Check resistances between the ground pins with an ohmmeter to confirm that all "ground" pins are indeed tied to ground. Now, install resistor networks RN1 and RN2 along with U1 and wire up all the signals involving these devices, right up to and including the DDS chip, U2. In the breadboard version, I wired up diagnostic LEDs to the outputs—a great aid for software development and for verifying that the right LPT port is being used to "talk" to the board. Power up the board and probe U2 pins 9, 10, 12, 13, 14, and 17—all of which should be a logic LOW. U2 pins 11 and 18 should be HIGH. Short the following DB-25 pins one by one to ground and look for these responses:

DB25, pin 1	U2, pin 9 goes HIGH
DB25, pin 2	U2, pin 13 goes HIGH
DB25, pin 6	U2, pin 14 goes HIGH
DB25, pin 7	U2, pin 17 goes HIGH
DB25, pin 8	U2, pin 12 goes HIGH
DB25, pin 9	U2, pin 10 goes HIGH

The next phase requires an HF receiver and a PC running the Julieboard driver software. Wire up the oscillator module and install U2. (Bear in mind that the DDS chip is specified as being ESD-sensitive by Harris and can be damaged or even destroyed by improper handling. If possible, handle this chip only at a properly equipped

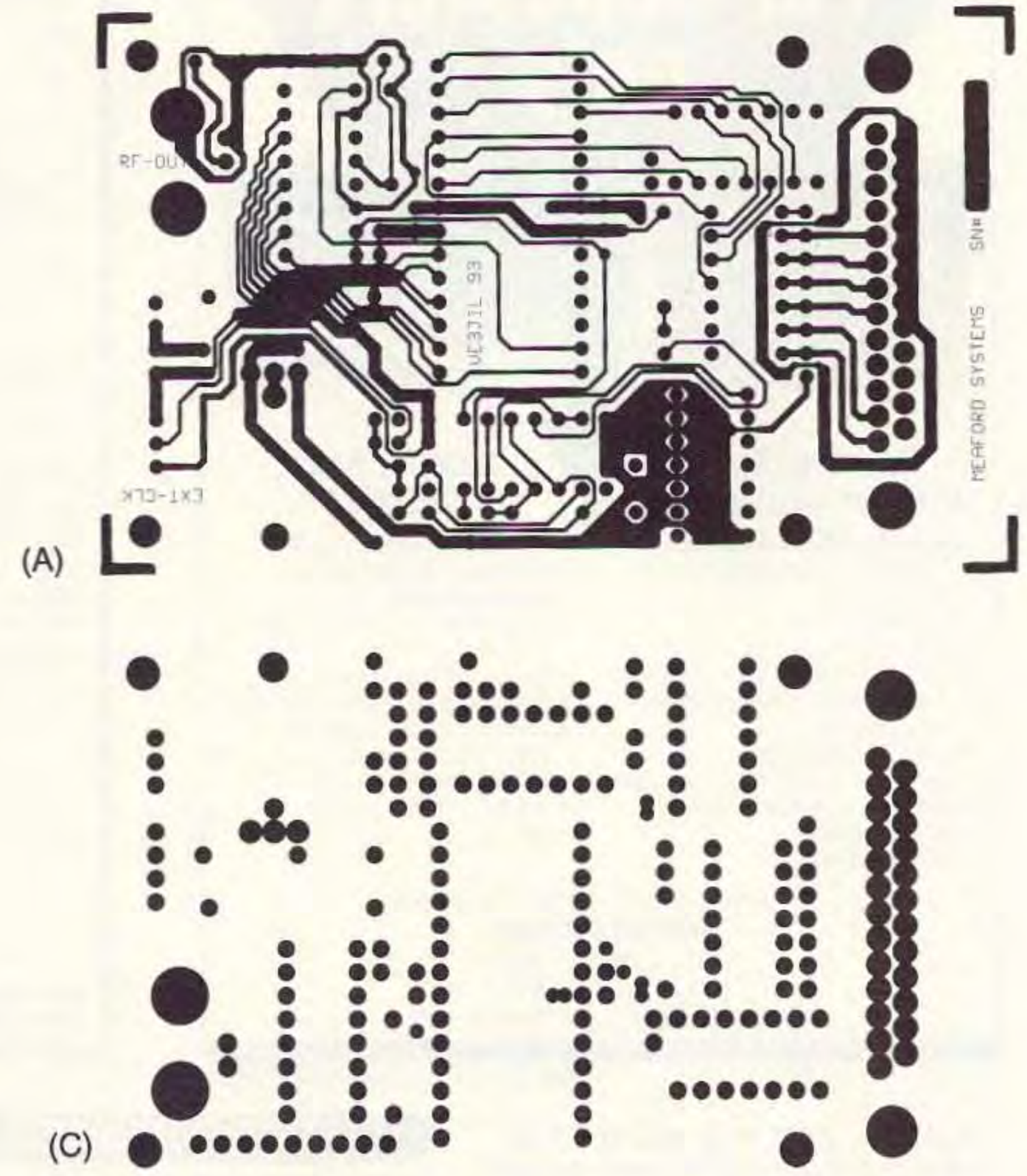
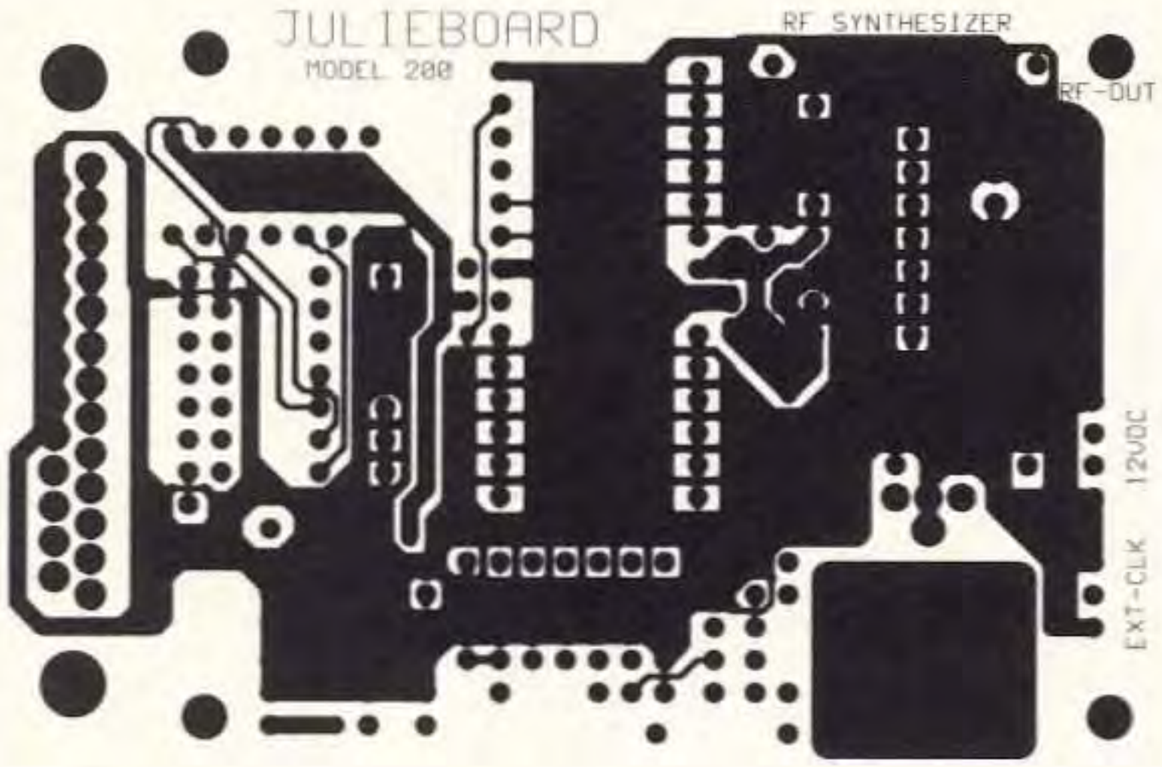
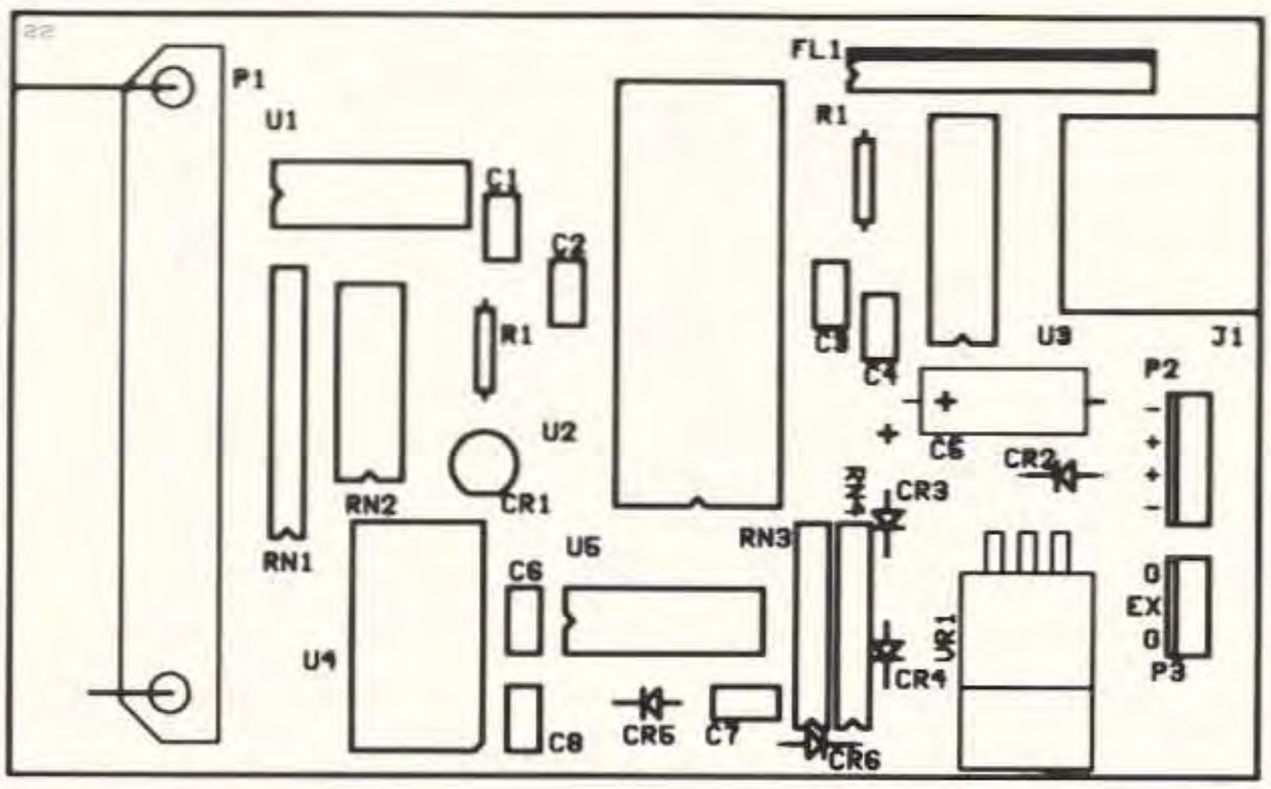


Figure 2. A) PC top foil pattern; B) PC bottom foil pattern;



(B)



(D)

C) PC drilling template; D) parts placement diagram.

ESD-protected workstation with wrist straps and an anti-static worktop.)

The 74F132 NAND gate U5 and its associated parts may be installed now.

Connect the PC to the Julieboard DB25 female connector via an appropriate cable (male DB25/male DB25 straight-through) and power up the board. Select a test frequency (this isn't critical, any frequency between 1 MHz and 16 MHz will do) and tune the receiver (CW or SSB mode) to that frequency. A clear continuous carrier should be audible fairly close to the expected frequency. (Use a short piece of wire in close proximity, but not touching, the board as the antenna for the receiver). Try tuning the signal in 100 Hz increments and listen for the corresponding changes in pitch. If it works . . . congratulations, you're almost there! If not, look for activity on U2 output pins 27-28 and 1-6: If they are completely dead, try verifying the presence of the 40 MHz TTL clock at U2 pin 16 and confirm proper U2 hookup. Check that:

VCC	pin 22 = +5V
VCC	pin 8 = +5V
BANKSEL	pin 9 = LOW
ENPHACC	pin 12 = LOW
LOAD*	pin 18 = HIGH
GND	pin 7 = GND
GND	pin 15 = GND
GND	pin 21 = GND

Try feeling the case of the DDS chip. If it is very hot, the problem likely involves the DDS chip itself; if it is stone cold, the problem could either be a faulty clock module or a dead DDS chip. A normally working DDS chip should be slightly warm—if this is the case, suspect a problem with the programming or control process. (Also check to see if it is getting +5V!).

Once DDS chip operation has been verified, wire and populate the CA3338 video DAC (U3) and the filter module (FL1) sockets, observing the same ESD precautions as for U2. The synthesizer output should look like a perfect sine wave, except for low frequencies which will show some "staircasing," courtesy of the D/A conversion process. Finally, verify the EXTERNAL CLOCK function with an external clock—if it works OK, then the construction of the synthesizer is complete. Have fun with the new toy!

Conclusion

I have had computer-controlled DDS synthesizers in my shack for several years now and would almost rather give up my scope or multimeter than do without them—they were well worth the development cost. So far, they have been used for:

- Frequency spotting
- Software-controlled VFO
- VFO for home-brew direct conversion receivers
- Digital retro-fit to analog equipment
- Remote tuning of transmitters/receivers
- Frequency-hopping/spread spectrum work
- Programmable secondary frequency standard
- Crystal/crystal-filter characterization
- Crystal oscillator substitution
- ATE signal generator/sweeper

I found that direct conversion receiver circuits worked especially well with this oscillator—tuning via software on the computer screen was a real novelty and the sound was particularly crisp and clear. Perhaps the next challenge will be a home-brew digital transceiver!

Personal Note

Why "Julieboard"? Well, back when I did my first DDS-for-a-PC design (this is my third), I needed a name for the project. At the time, my second-oldest daughter, Julie, was in her active toddler phase and the name seemed appropriate for a board originally intended for frequency hopping development work. Since I have three other daughters, I suppose I'll have to do at least three other boards so that Julie's sisters don't feel left out! 73

Continued on page 46

Parts List

Quantity	Part	Description	Digikey #
1	U1	74HC14	MM74HC14N
1	U2	Harris HSP45102	PC-40
1	U3	Harris CA3338AE	CA3338AE
1	U4	40.000 MHz osc. module	CTX120
1	U5	74F132 (can sub 74F00)	(74F00PC)
1	VR1	7805 regulator (TO-220)	AN7805
1	FL1	Coilcraft filter module	K9686-5
1	CR1	Green light emitting diode	P309
2	CR2,CR6	1N4001 diode	1N4001GI
3	CR3-5	1N914 diode	1N914APH
1	R1	1K5 5% 1/4W resistor	1.5KQ
1	R2	75R 5% 1/4W resistor	75Q
1	RN1	47k resistor network (8sip7)	Q7473
1	RN2	100R resistor network (14dip7)	760-3-R100
1	RN3	1K5 resistor network (8sip4)	Q4152
1	RN4	100R resistor network (8sip4)	Q4101
7	C1-4,C6-8	100N ceramic cap (.1" L.S.)	P4917
1	C5	1U0 aluminum electrolytic cap	P1345
1	J1	Right angle BNC connector	Mouser #177-3138
1	P1	Female DB25 right-angle connector	425F-ND
1	P2	4x1 male header	WM4202
1	P3	3x1 male header	WM4201
1		Blank printed circuit board	
1		heat sink	#HS106-ND
1		thermal compound	
1		male/male DB25 cable	C7MMT-251OG-ND

Note: J1 is AMP # 228686-1
P1 is AMP # 745353-4

Optional:

1	U2	28p machined (gold) IC socket	AE7228
1	U3	16p machined (gold) IC socket	AE7216
3	U1,U4-5	14p machined (gold) IC socket	AE7214

Blank printed circuit boards, partial kits, and finished units are available from the author (Box 232, Pakenham, Ontario, CANADA K0A 2X0; (613) 624-5247).

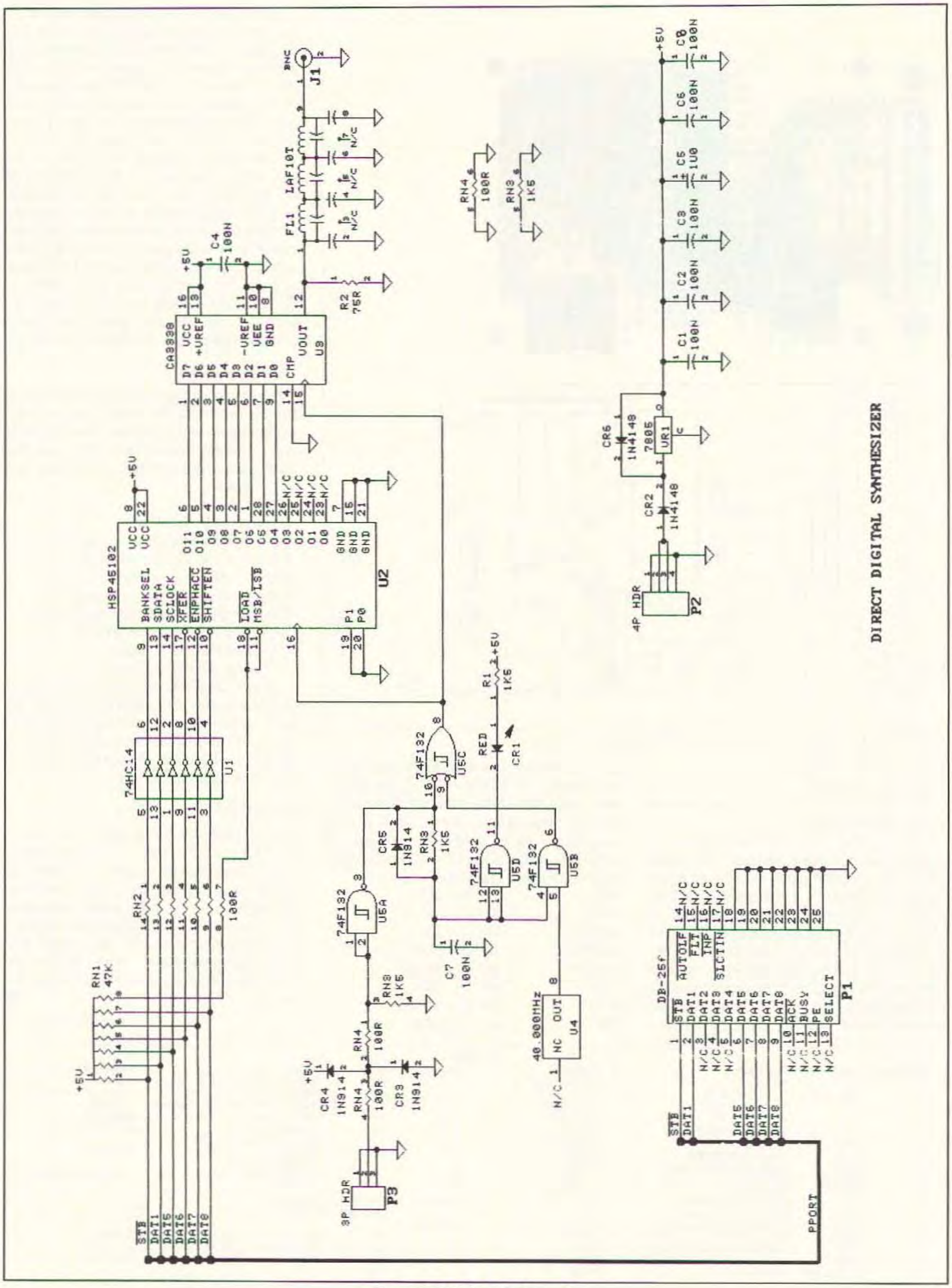


Figure 3. Julieboard schematic.