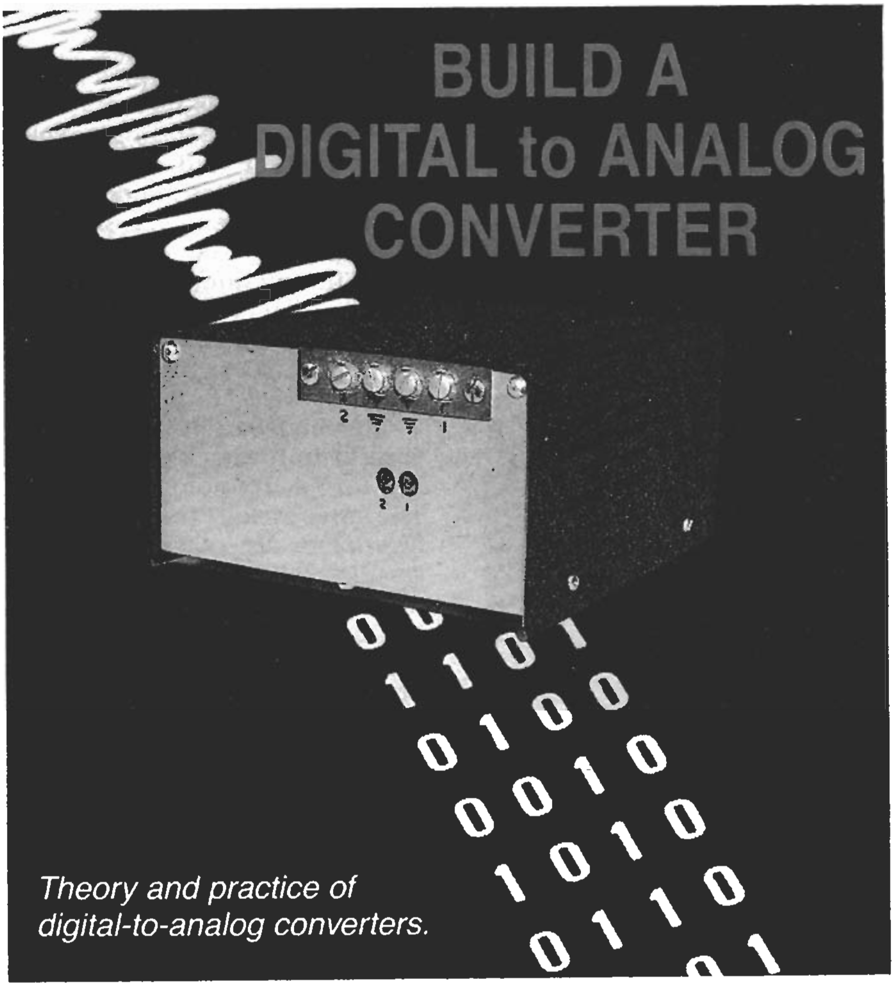
BUILD A DIGITAL to ANALOG CONVERTER



Theory and practice of digital-to-analog converters.

DAVID WEBER

Computers are digital devices. In a computer, all information can ultimately be resolved into bits that are either on or off. The real world, however, is analog. The sun rises gradually, not suddenly. Temperature changes smoothly. Water flows. All human sensory inputs receive data in analog format.

However, it's difficult to represent analog values digitally. When people began to design thinking machines, they quickly discovered that devices that operated on analog principles were complicated, sensitive, and unreliable. The problem needed simplification; what greater simplification than to allow no more than two states, on and off?

The problem, of course, is that operation on digital principles erects a fundamental barrier between the computer and the world outside the box. How can the computer, a yes/no device, sense, much less control, analog phenomena in the real world?

Two complementary devices make both sensing and control possible: the *A*nalog-to-*D*igital *C*onverter (ADC) and the *D*igital-to-*A*nalog *C*onverter (DAC), respectively. As the name suggests, an ADC is a device that converts real-world analog quantities into digital terms that the computer can deal with. Conversely, a DAC

continued on page 81

D/A CONVERTER

continued from page 77

is a device that converts the computer's digital data to analog form, usually current or voltage.

In this article we'll show what a DAC is, how it works, and how it can be used. To illustrate those ideas, we'll present construction details for building a low-cost ($35), high-speed DAC, with as many as eight independent channels. You can connect the DAC to any standard parallel printer port. We'll also discuss the software that controls the DAC, but space precludes printing full listings. However, the software (DAC.ARC) is available on the R-E BBS (516-293-2283). A special feature of the software is an interactive full-screen editor/compiler that functions much like Borland's Turbo and Microsoft's Quick languages.

DAC basics

The most common technique for making an analog signal from a digital value is the R-2R resistor ladder; an eight-stage ladder is shown in Fig. 1. A precision reference voltage (V_{REF}) is applied to the top of the ladder; current then passes through the R-2R resistor network to ground. The strength of the output current (I_{OUT}) is determined by which of the S1 through S8 switches are open or closed.

The maximum I_{OUT} occurs when all the switches are closed, which places one end of all the 2R resistors at ground potential (disregarding the low resistance of ammeter M1). Solving for the equivalent resistance we get R, so the maximum full-scale current is V_{REF}/R. On the other hand, if only S1 is closed (S2–S8 open), then the maximum current is V_{REF}/2R. The current is now one-half the original full-scale current.

It should be obvious to you now that closing any combination of switches will yield a specific fraction of the full-scale current. By rapidly changing the digital word that closes and opens switches S1 through S8, a changing current will be present at I_{OUT}, and

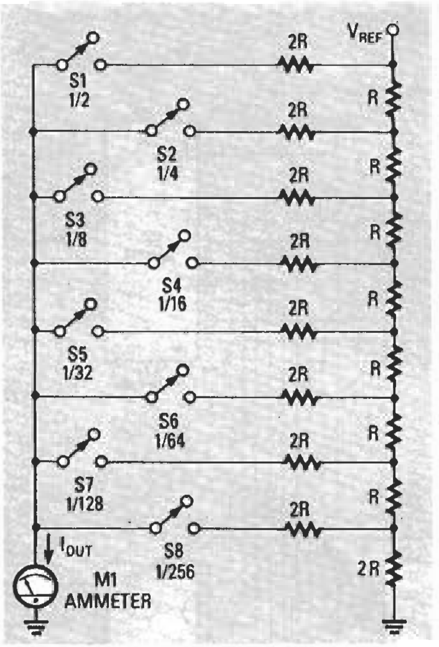


***Fig. 1.** THE SIMPLEST DAC: An R/2R resistor ladder. Each switch that is closed increases the amount of current at I_{OUT}.*

that's digital-to-analog conversion.

EMThe precision of I_{OUT} depends on the precision of V_{REF}, on the precision of the resistors in the ladder, and on the precision with which they are matched. In addition, the number of stages in the ladder determines the fineness with which I_{OUT} can be adjusted. Commercially, eight-bit DAC's are inexpensive and readily available; sixteen-bit DAC's are available at higher cost for high-precision applications.

The DAC0808

The hardware described in this article is built around an eight-bit DAC, the DAC0808 (also known as the LM1408). It has been around for some time, is a stable design, is widely available, and is inexpensive. Eight bits give a precision of 0.39%, which is more than adequate for many uses. The typical settling time for the DAC0808 is 150 ns; that translates to a switching speed better than 5 MHz. In fact, a standard PC cannot come close to driving the converter at that speed. (A 35-kHz square wave was generated using a hand-optimized assembly-language loop on a 4-MHz CPM machine directly driving an 8255 port.)

The internal construction of the DAC0808 differs somewhat from the idealized resistor ladder discussed here. Output is provided on two pins as complementary currents, rather than on a single output as shown in Fig. 1. Also, the lower four stages of the ladder are driven separately from the upper four, because the lower stages are more sensitive to error. The fundamental operating principles, however, are the same.

The circuit

The schematic for the circuit is shown in two parts. The first section (shown in Fig. 2) details the digital interface to the PC's parallel port. One analog section is shown in Fig. 3; bear in mind that the digital section can drive eight analog sections.

An eight-bit parallel printer port is an ideal interface for a DAC, because it is typically the fastest standard interface on a PC. The Centronics parallel port has a 36-pin connector, of which the IBM PC family uses 25; our DAC uses only 16 of those. Eight lines are for data, one is for ground, and the remaining seven are for status and handshaking.

Let's discuss the status lines (PE, BUSY, SELECT, and $\overline{\text{ERROR}}$) first. When power is applied to the interface, the signal levels on the PE (paper empty) and BUSY lines are low, and the levels on the $\overline{\text{ERROR}}$ and SELECT lines are pulled high by R1 and R2. Those levels prevent the computer from hanging up, thinking that a printer was out of paper, or busy, or in an error condition. We're able to ignore the busy line because the DAC is so much faster than the computer.

When power is off, $\overline{\text{ERROR}}$ and SELECT go low, thereby indicating to the computer that the DAC is in an error state and not selected. In that way the computer can sense whether or not the DAC is powered and ready for data.

There are three handshaking signals: $\overline{\text{RESET}}$, $\overline{\text{STROBE}}$, and $\overline{\text{ACK}}$. $\overline{\text{RESET}}$ is normally used to reset a printer. In our circuit, however, $\overline{\text{RESET}}$ is used to load the value that defines which of the eight converters will receive the data that follows.

The $\overline{\text{STROBE}}$ and the $\overline{\text{ACK}}$ lines handshake data through the par-

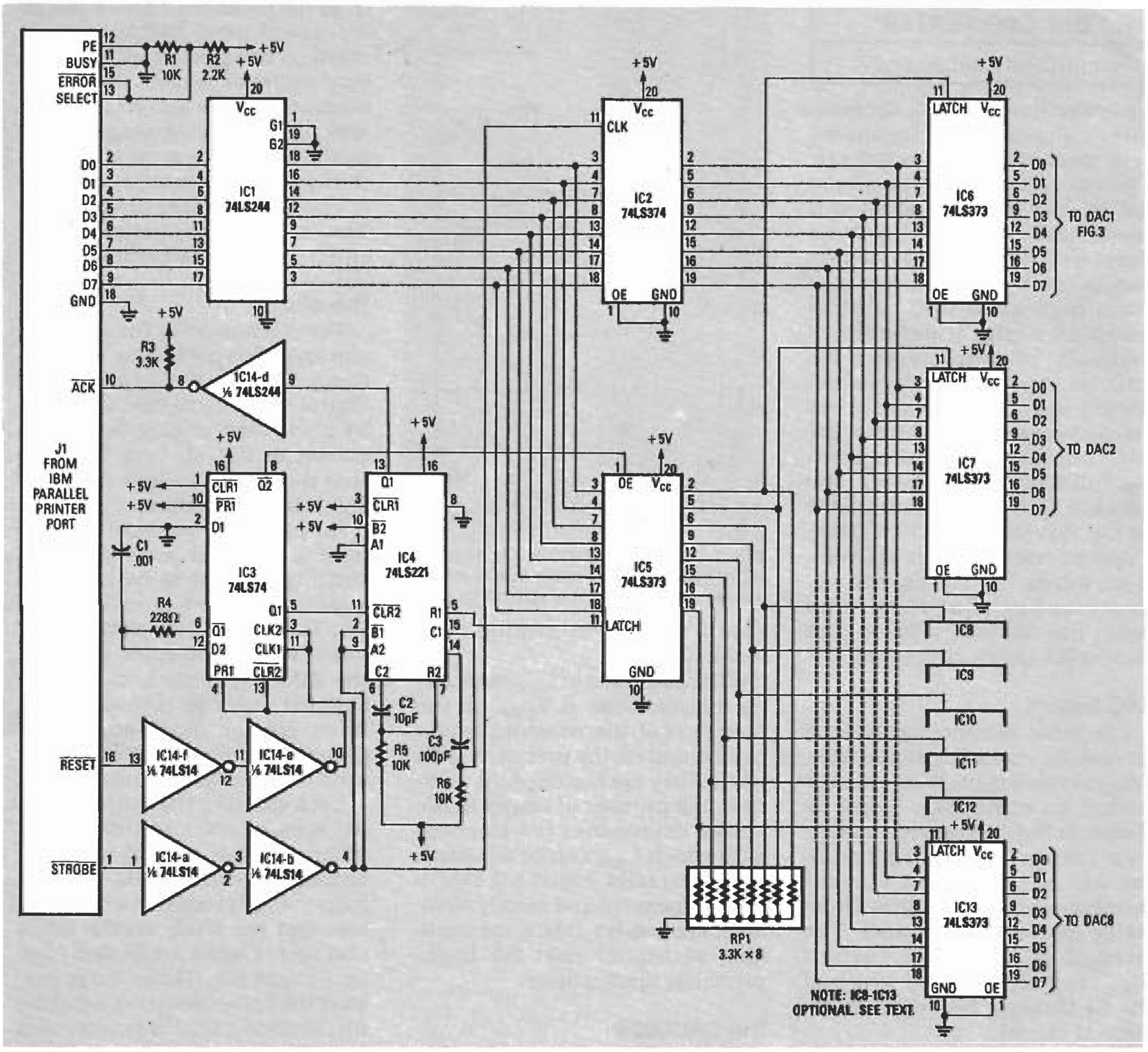


Fig. 2. *DIGITAL SECTION: A standard parallel printer port can control eight digital latches, for a total of 64 bits of digital I/O.*

allel port pretty much as they would for a printer. The computer puts a byte of data on D0–D7 and pulses $\overline{\text{STROBE}}$ to tell the parallel device that data is ready. The parallel device reads the data and responds with $\overline{\text{ACK}}$ to signal that everything is fine. To understand what happens in detail, let's walk through a typical data-transfer sequence. Assume that one analog section has already been selected (we'll show how that's done momentarily).

A byte of data enters the circuit and is buffered by IC1, an octal data buffer; IC1 cleans up the signals after their long journey down the cable. The data then proceeds to IC2, an octal data latch; IC2 will latch the data when it gets $\overline{\text{STROBE}}$ from the computer. As for $\overline{\text{STROBE}}$, it is conditioned by a pair of 74LS14 inverters, IC14-a and IC14-b. It is the rising edge of $\overline{\text{STROBE}}$ that latches the data in IC2.

The output of IC2 is presented to latches IC6–IC13; the latter are what drive the eight converters. Which latch picks up the data depends on the mask stored in IC5, but we'll get to that in a moment. For now, let's look a little closer at what $\overline{\text{STROBE}}$ does.

In addition to latching the data into IC2, $\overline{\text{STROBE}}$ drives a dual one-shot multivibrator, IC4, a 74LS221, which generates a pulse of precisely controlled width—1 µs, in this case. After buffering by IC14-d, that pulse becomes the $\overline{\text{ACK}}$ signal that tells the computer the transaction is complete.

Now let's see how the data latches (IC6–IC13) are selected by IC5, also a 74LS373. When one of IC5's outputs is high, the associated latch will be selected and its outputs will follow the data stream presented to the device. When an output is low, the latch ignores the data stream. In addition, the last value sent (when the latch enable signal was high) is retained in IC5's outputs.

Because of that arrangement,

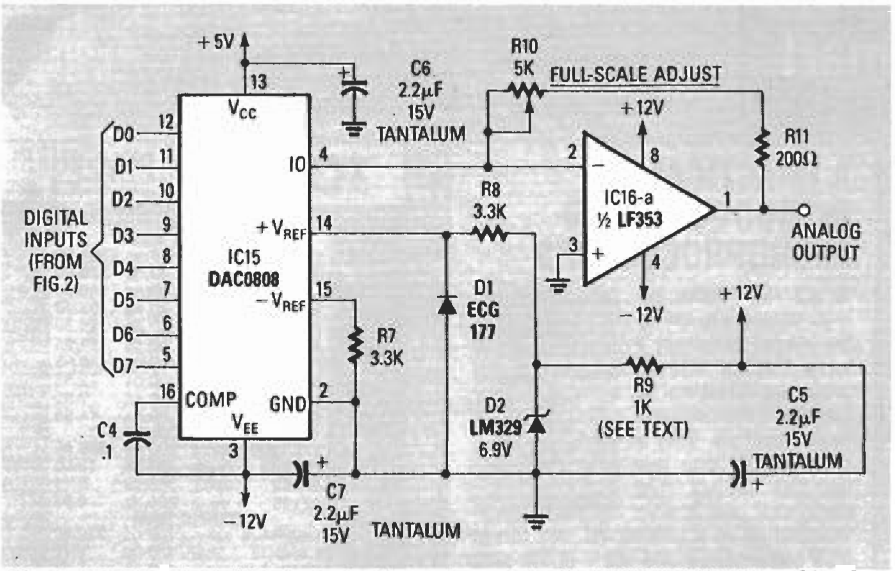


Fig. 3. *ANALOG SECTION: A single channel consists of a DAC0808 (IC15) and half an op-amp (IC16-a). The voltage reference (D2) is common to all channels, but the value of the dropping resistor (R9) varies as the number of DAC's installed in the system.*

the same data may be written to several latches simultaneously. For example, a mask of 0Ch (0000 1100) would enable latches three and four and disable the remaining latches. Subsequently, any data written to the device would put the same data in both latches three and four.

How is the selection mask loaded into IC5? Via the PC's $\overline{\text{RESET}}$ line. When $\overline{\text{RESET}}$ goes low, section two of IC3, a 74LS74 dual flip-flop, is cleared, and that causes IC5's output enable ($\overline{\text{OE}}$) line to go high. That in turn causes IC5's outputs to go into a high-impedance state, so the latch buffers (IC6–IC13) retain their current contents on their outputs. Also, those latches will ignore the next byte of data (which will be the latch-selection mask) from the PC.

Now the PC sends that selection byte; the $\overline{\text{STROBE}}$ line is pulsed, and that toggles the other half of flip-flop IC3 and also causes IC4 to send a 100-ns pulse to selection register IC5. That pulse latches the selection mask data into the selection register. The selection mask is now where it should be and has been kept out of the places where it was not wanted.

When the next data byte comes, $\overline{\text{STROBE}}$ is toggled again, so IC3 returns control of the selection lines to IC5, which now has a new mask. The data is loaded into the converter(s) specified by the new mask.

The purpose of R4 and C1 (between the two halves of IC3) is to guarantee that the second gate is not flipped until two $\overline{\text{STROBE}}$ pulses have been sensed. The RC combination delays the response to the first $\overline{\text{STROBE}}$ for about 100 ns, leaving it ready to sense the second $\overline{\text{STROBE}}$. Because the time delay is a function of the output drive of $\overline{\text{Q1}}$ and the input impedance of D2, substitutions should not be made for functionally equivalent IC's (74HC74, 74S74, 74F74, etc.) without verifying that the time constant remains the same, or using a different resistor or capacitor to maintain the time constant.

The analog circuit

Figure 3 shows one DAC channel. The digital circuitry can handle as many as eight DAC channels, but there's no need, of course, to build more channels than you need. However, because the analog amplifiers (IC16, an LF353) contains two amplifiers per package, it makes sense to add DAC's and associated components in pairs.

In the diagram, there are three fundamental components: the DAC, a voltage reference, and an amplifier. The DAC takes the digital number from the latch and the reference voltage from the LM329, and generates a proportional output current that drives the amplifier. Let's look at each component in detail.

The LM329 is a precision temperature-compensated Zener voltage reference. It creates a 6.9-volt source with a long-term stability of 0.002% and a temperature sensitivity of 0.01% per degree Centigrade. Only one LM329 is needed, regardless of the number of converters. However, the dropping resistor (R9) between the LM329 and the +12-

Parts List

Resistors

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

R1, R5, R6	*10,000 ohms*
R2	*2200 ohms*
R3, R7, R8	*3300 ohms*
R4	*220 ohms*
R9	*1000 ohms (for two DAC's; see text)*
R10	*5000 ohms, ten-turn potentiometer*
R11	*200 ohms*
RP1	*3300 ohms, 8 positions, common ground*

Capacitors

C1	*0.001 μF ceramic capacitor*
C2	*10 pf ceramic capacitor*
C3	*100 pf ceramic capacitor*
C4	*0.1 μF ceramic capacitor*
C5–C7	*2.2 μF, 15 volts, tantalum bypass capacitor*

Semiconductors

IC1	*74LS244 octal buffer*
IC2	*74LS374 octal latch*
IC3	*74LS74 dual D flip-flop*
IC4	*74LS221 dual monostable multivibrator*
IC5–IC7	*74LS373 octal latch*
IC8–IC13	*74LS373 octal latch (optional)*
IC14	*74LS14 hex Schmitt trigger*
IC15	*DAC0808 8-bit D/A converter*
IC16	*LF353 high-speed op-amp*
D1	*ECG177 clamping diode*
D2	*LM329 6.9-volt precision Zener*

Miscellaneous

Perf board, power supply, IC sockets, wire, solder, etc.

volt power supply should be adjusted for various loads. With two DAC's it should be 1000 ohms, with four it should be 510 ohms, with six, 330 ohms, and with eight, 240 ohms. In addition, even though only one dropping resistor and LM329 reference are needed for all eight converters, the 3.3K resistor (R8) and clamping diode (D1) must be duplicated for each converter.

The op-amp is a high-speed, high input-impedance, low power consumption model. It does not have a lot of output drive, but it can produce 10.0 volts across a 2K load. Its strong points are that it is very precise, that it has no offset voltage, and that it can change voltage at the maximum speed of the DAC. The feedback potentiometer (R10) on the amplifier allows you to adjust the full-scale output from 0.5 to 10.0 volts. You'll want to use a 15-turn potentiometer to make fine adjustments easily.

Construction

Unlike some projects, construction details are important. One of the most common failings of mixed analog/digital designs is noisy digital signals corrupting sensitive analog devices. So keep the analog section physically separate from the digital section, as shown in Fig. 4. In addition, keep the analog lead lines short, and make sure that the feedback resistor on the LF353 amplifier is placed close to the IC and far from the digital parts.

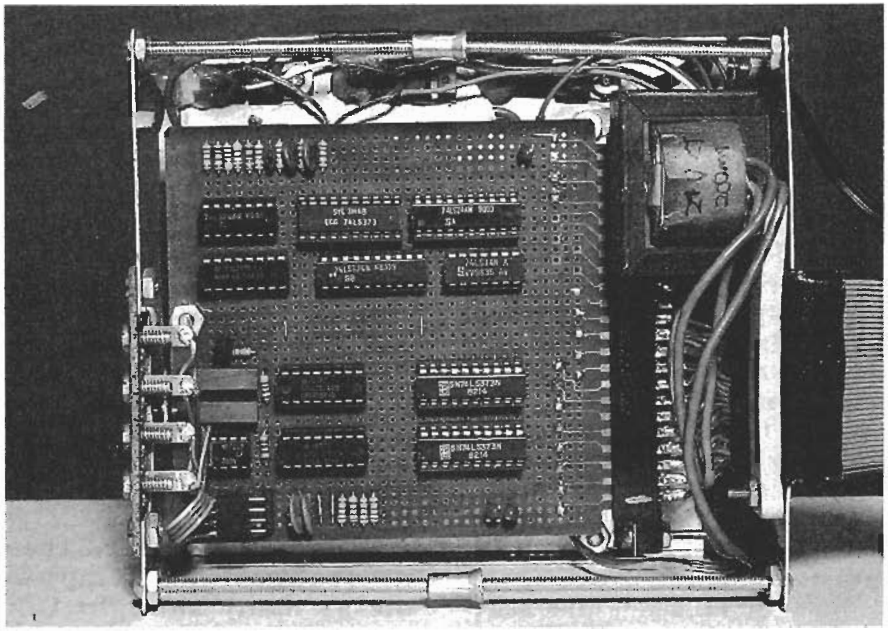


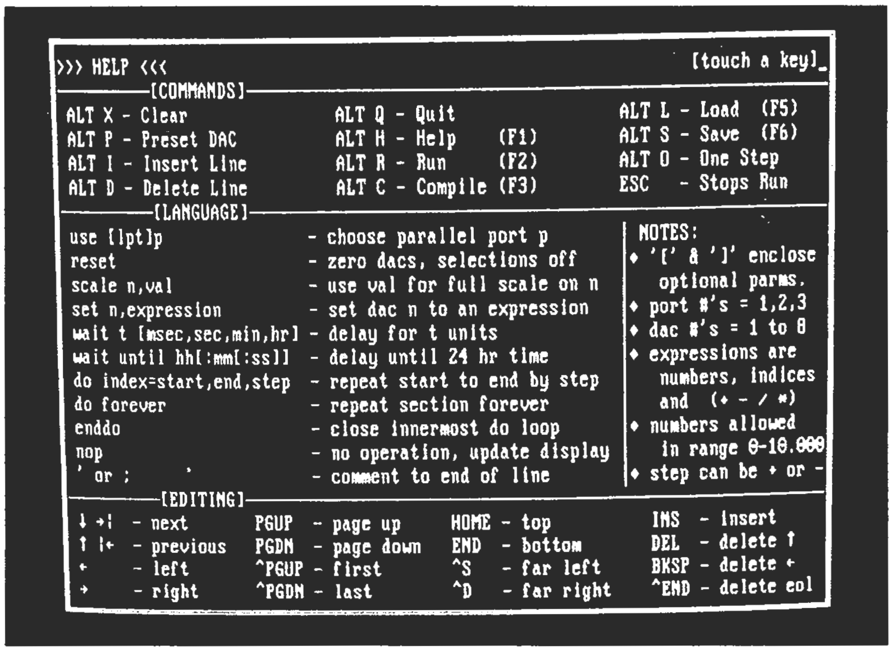
Fig. 4. *SEPARATE THE ANALOG AND DIGITAL SECTIONS of the circuit. The author used point-to-point wiring.*

Some components can be substituted. A 74LS123 can replace IC4, the 74LS221. Except for IC3, each of the other digital parts can use the corresponding member of another family. For example, 74373, 74C373, 74S373, 74F373, 74HC373, 74HCT373, 74ALS373. Also, an LM1408-8 is a direct replacement for the DAC0808.

The maximum power required when driving eight DAC's into full loads is +5 volts at 250 mA, +12 volts at 100 mA, and −12 volts at 100 mA. Of course, the supply voltages must be regulated too. Power-supply bypassing is also a must. Put a 2.2-µF tantalum capacitor (C5–C7) between ground and each of the three power supplies.

TABLE 1—DACL PROGRAM STATEMENTS

Statement	Meaning
' or ;	The rest of the line is a comment.
use [1pt]p	Use parallel port 'p' for the converter.
reset	Reset the converters on the current port to 0.00
scale, n,val	Use 'val' as a full scale value for converter 'n'
set n,exp	Set value of converter 'n' to expression.
wait t [msec, sec,min,hr]	If no units given, milliseconds assumed
wait until hh [:mm[:ss]]	Wait until 24 hour time.
do index= start, stop, step	Loop on index from start to stop by step.
do forever	Do forever or until user presses ESC
enddo	Close innermost do loop
nop	Do nothing except update display.



```
>>> HELP <<<                                                  [touch a key]_
──────────[COMMANDS]──────────────────────────────────────────────────────────
 ALT X - Clear             ALT Q - Quit                ALT L - Load  (F5)
 ALT P - Preset DAC        ALT H - Help     (F1)       ALT S - Save  (F6)
 ALT I - Insert Line       ALT R - Run      (F2)       ALT O - One Step
 ALT D - Delete Line       ALT C - Compile  (F3)       ESC   - Stops Run
──────────[LANGUAGE]──────────────────────────────────────────────────────────
 use [lpt]p                  - choose parallel port p       │ NOTES:
 reset                       - zero dacs, selections off    │♦ '[' & ']' enclose
 scale n,val                 - use val for full scale on n  │   optional parms.
 set n,expression            - set dac n to an expression   │♦ port #'s = 1,2,3
 wait t [msec,sec,min,hr]    - delay for t units            │♦ dac #'s = 1 to 8
 wait until hh[:mm[:ss]]     - delay until 24 hr time       │♦ expressions are
 do index=start,end,step     - repeat start to end by step  │   numbers, indices
 do forever                  - repeat section forever       │   and  (+ - / *)
 enddo                       - close innermost do loop      │♦ numbers allowed
 nop                         - no operation, update display │   in range 0-10.000
 ' or ;                      - comment to end of line       │♦ step can be + or -
──────────[EDITING]───────────────────────────────────────────────────────────
 ↓ →|  - next      PGUP  - page up     HOME - top         INS  - insert
 ↑ |←  - previous  PGDN  - page down   END  - bottom      DEL  - delete ↑
 ←     - left      ^PGUP - first       ^S   - far left    BKSP - delete ←
 →     - right     ^PGDN - last        ^D   - far right   ^END - delete eol
```

Fig. 5. *THE DACL EDITOR/COMPILER. Compilation is nearly instantaneous; programs may be single-stepped.*

DAC, SCHMACK

You may think that DAC's are used only in scientific laboratories and industrial controls. Actually, they're starting to show up in everything from hospital equipment to automobiles to the PC's we all use. For example, the latest display adapter for the IBM family, VGA, uses a DAC for representing the red, blue, and green signals that drive the CRT. All previous IBM display adapters were digital in nature; they were at best able to display as many as 64 colors. The VGA, by contrast, can display as many as 256,000 colors, because each color line can take on many values.

A DAC is very useful for generating test waveforms for hardware development, debugging and troubleshooting, and the like. Even something as trivial as controlling the intensity of a light show is amenable to a DAC.

Now calibrate the system. Unplug the device from the computer's parallel port and turn it on. Attach a voltmeter to the output terminals of the first DAC, and adjust it for the desired full-scale value. Repeat the adjustments on the second DAC and continue until all have been set.

Use

It's easy to send data and selection masks to the converter. Loading a selection mask takes three steps: First, pulse the $\overline{\text{RESET}}$ line low, then high. Second, load the selection mask on the data lines. Last, pulse the $\overline{\text{STROBE}}$ line low then high. Sending data to the converter is a two-step process. First, load the data on the lines. Second, pulse the $\overline{\text{STROBE}}$ line low, then high.

Unfortunately, we don't have space to present program listings here. However, sample driver programs are included in DAC.ARC, which is available on the R-E BBS (516-293-2283). Two drivers are provided; DACGEN.ASM and DACPAR.ASM. DACPAR.ASM controls the parallel port directly. It is the fastest method, but requires close compatibility with the IBM standard. The other driver, DACGEN.ASM, uses the BIOS to control the parallel port. DACGEN.ASM should run on any IBM compatible, but it will transfer data at a slower rate because of the BIOS overhead.

Functionally, the drivers are identical. Each provides three function calls: one to obtain the status of the DAC; one to send a selection mask to the DAC; and one to send data to the DAC.

In addition, the author has developed a special programming language called DACL (*D*igital to *A*nalog *C*ontrol *L*anguage), and an integrated programming environment that allows you to experiment with programming the hardware. DACL's program statements are listed in Table 1.

The environment is illustrated in Fig. 5. With it, program development is as easy as in an interpreted language like BASIC; but run-time speed rivals that of a compiler. As shown, a full-screen editor is on the left, and current DAC status is shown on the right. Help is available by pressing Alt-H. The compiler produces plain English error messages, and leaves the editor pointing to the offending line. Debugging tools permit single-stepping through the source program. DACL comes in two versions, generic MS-DOS and IBM compatible. The first offers hardware generality, the second provides speed of execution. Sample programs and a manual are provided in DAC.ARC. ◆CD◆