60¢ ■ FEB. 1974

Radio-Electronics®

FOR MEN WITH IDEAS IN ELECTRONICS

The Problems & The Promise Of
DISCRETE CD-4 RECORDS

EXCLUSIVE TO R-E
Heathkit Builds
A New Kind Of
Color TV Kit

Tips For Experimenters
IC's FOR ELECTRONIC MUSIC

Inductors Without Coils
ALL ABOUT IC GYRATORS

IC Memories Are Fun
WHAT IS A ROM?

PLUS

Appliance Clinic

J

R Guide

St Troubleshooting

A GERNSBACK PUBLICATION

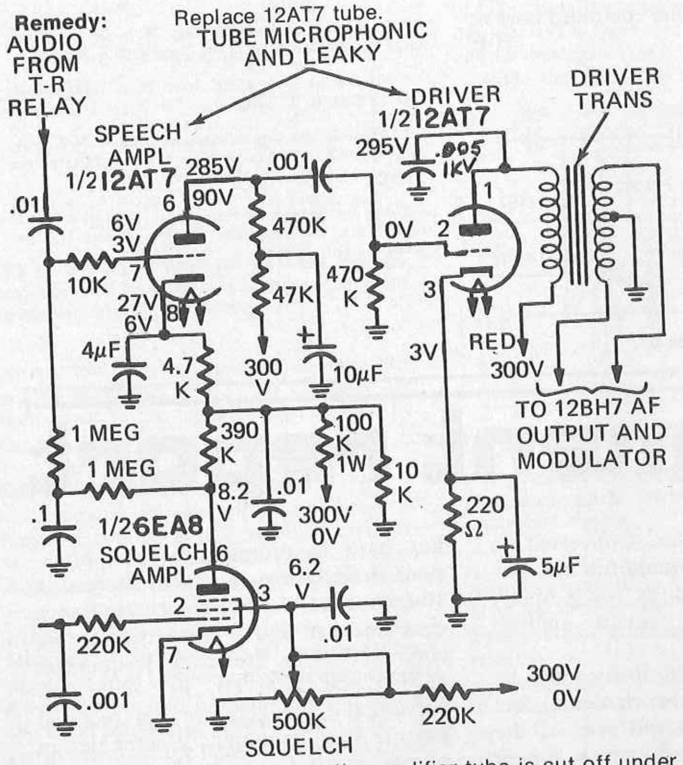
CB CASEBOOK

by ANDREW J. MUELLER

Case 1: Speaker crackles when unit is squelched.

Common to: Sonar **FS-23**

Remedy: Replace 12AT7 tube.

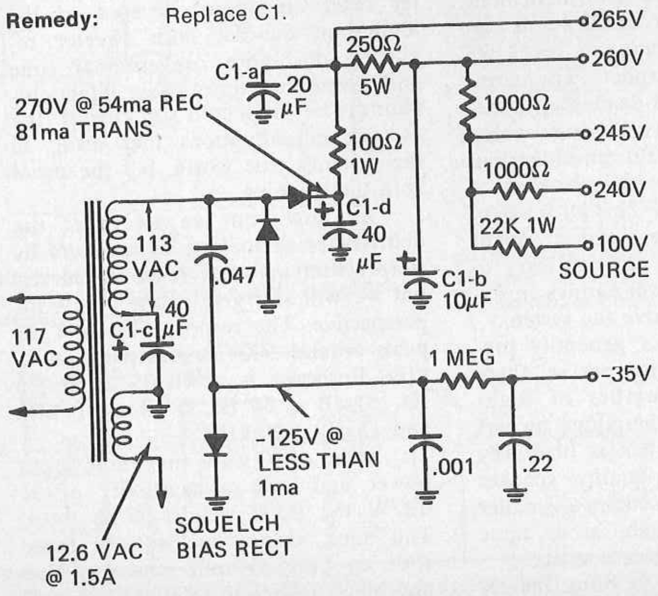


Reasoning: When the audio amplifier tube is cut off under squelched conditions, any noise due to microphonics, shorts, leakage, etc., that is generated can be clearly heard through the speaker. The tube was checked and found to be OK but direct replacement solved the trouble.

Case 2: Radio lights up the does not transmit or receive.

Common to: Metroteck **Mustang**

Remedy: Replace C1.

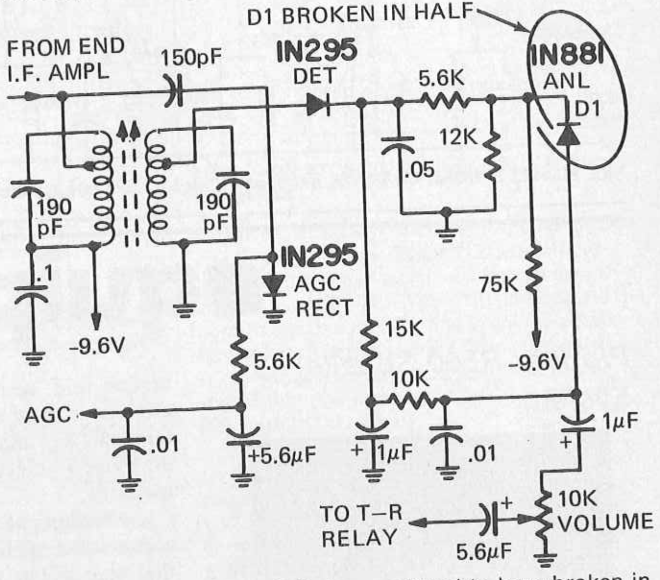


Reasoning: C1, the power supply electrolytic was found dried out. This caused the power supply to deliver only one half of the rated voltage. Replacement of C1 restored the unit to normal operation.

Case 3: Receive is weak and distorted.

Common to: Jonson **110**

Remedy: Replace D1.

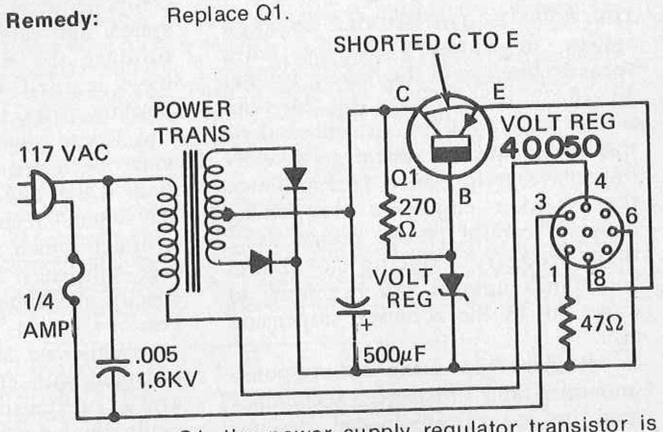


Reasoning: D1, the anl diode was found to have broken in half due to fatigue. This broke the path of receiver audio from the detector to the af amplifier. A little audio did feed through due to circuit capacitance but it was distorted.

Case 4: Unit hums on receive and transmit when operated from 117 vac.

Common to: Heathkit **GW-14.**

Remedy: Replace Q1.



Reasoning: Q1, the power supply regulator transistor is shorted. This feeds about 25 volts instead of 12 volts to the radio. In addition, there is very little filtering being done so all of the power supply ripple is fed to the radio. This results in the ac hum that is heard on both transmit and receive.

IC's for electronic music

Electronic music is a fast-growing field with the synthesizer arousing the most interest among the avant-garde. Regardless of what you want in electronic music, IC's will simplify design and ease construction.

SUPPOSE YOU WERE GOING TO DESIGN AND build an electronic music synthesizer, a pitch reference, an electronic organ, a composer, a timbre generator, or some entirely new instrument. What devices would you use? Where would you go for help?

While there are a few integrated circuits that are obviously and specifically intended for music use, these are rare, usually expensive in small quantities, and often hard to impossible to get. On the other hand, there are great heaping piles of different IC's available that don't even hint they are good for music use. These, or at least some of them, are widely available, cheap, and, best of all, many of the latest dramatically simplify things, doing as good and sometimes much better a job than older circuits did. In fact, some circuits are now available that are almost hard to believe—a very stable sine, square, triangle VCO for $3, a *hex* voltage-controlled amplifier for $1.80; a tracking "glider" phase-lock-loop that works over a 2000:1 frequency range without harmonic locking and costs under $5; a *single IC* to generate all the equally tempered notes of one octave; and switches that handle analog or digital, one to N or N to one reversibly, for under $2.

Here's my selection of a few dozen or so integrated circuits that are (1) cheap, (2) widely available, (3) applicable to electronic music, and (4) do a job far simpler or cheaper than older approaches. Table 1 lists all the manufacturers and their addresses. All prices are approximate. Be sure to have good data sheets and application notes on hand before you try to use *any* integrated circuit.

One IC top octave generator

Most music is arranged into twelve-note *equally tempered* note groupings (take a look at a piano keyboard). As you go up in frequency one *octave*, you double frequency on the *thirteenth* note. The note spacing is NOT linear, it is exponantial. Each note is spaced from its neighbor by $\sqrt[12]{2}$ or approximately 6%. There is no reasonable way to exactly generate anirrational number such as $\sqrt[12]{2}$, so you have to *approximate* it the best way you can. Usually you start with a 1 or 2 *megahertz* crystal and then divide down by some "magic" optimum series of numbers (often 239 - 253 - 268 - 284 - 301 - 319 - 338 - 358 - 379 - 402 - 426 - 451) to get a good enough approximation to the highest octave you care to generate. From here you pick up the rest of the notes with a simple string of binary dividers.

The circuitry that handles the top octave is called a *top-octave generator*. Many of these circuits had been based on the GEM555 and GEM556 a pair of hard-to-use, harder-to-get IC's that are now essentially obsolete.

The *Mostek* MK5024P/AA is a one-chip,

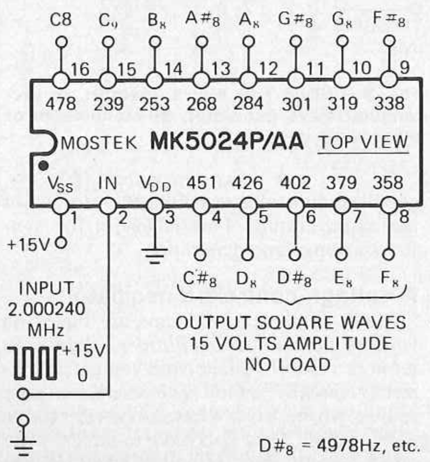
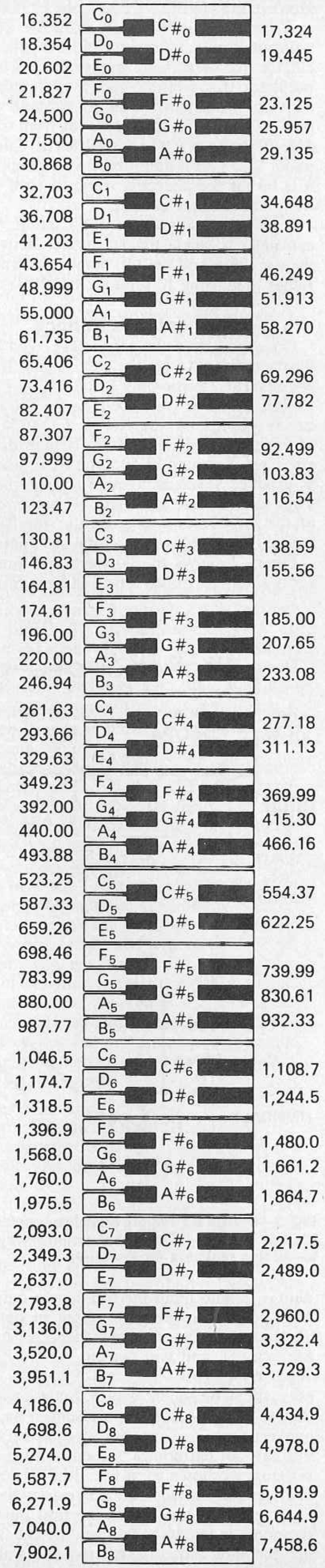


FIG. 1—SINGLE IC top octave synthesizer.



single 15-volt supply top-octave generator. Hook it up as in Fig. 1. You input a 2.000240 megahertz squarewave or sinewave of 15 volts amplitude, obtained from a crystal oscillator (for permanent tuning) or a variable oscillator (for vibrato, glides, or tuning to another instrument). You get thirteen outputs, appearing as square waves from C8 at 4186.01 hertz to C9 at 8369.2 hertz. Cost is under $12. This is admittedly a bit steep, but it is by far the cheapest route to go if you want all the notes at once.

If you only want one octave at a time, you can place a single binary divider between the oscillator and the top octave generator, rather than using 12 separate dividers.

Seven octaves at once

Once you have the top octave, you add binary dividers to get the rest. Again, there are several "music-only" divider IC's available, but none is as good, as easy to use, or as cheap as the RCA CD4024 or Motorola MC14024 CMOS 7-stage dividers. Cost is around $3.50. One IC is needed to produce seven octaves of a single note. Thus a top octave generator IC and twelve of the CD4024's will generate simultaneously all eight octaves or a total of 97 notes.

Figure 2 shows the connections. Simply apply a voltage from +3 to +18 (best results with +15) and input the top octave output. You get out seven new notes in octave steps. For instance, input A#8 (A sharp, 8th octave), and you get out A #7, A#6, A#5, A#4, A#3, A#2, and A#1. The outputs will all be square waves. Square waves only have odd harmonics present. You can convert these to sawtooth waveforms with virtually all harmonics present with a few resistors as shown as in Fig. 2-b. While the stairstep may not *look* quite like a sawtooth, analyze it and you'll find the first missing harmonic is the 16th, followed by the 32nd and the 48th, etc. . . . Otherwise it is absolutely identical to a linear ramp. Filtering is used to convert either the square or sawtooth outputs into familiar tone colors. For instance, the square waves are often used for clarinet and stopped organ sounds; the sawtooth by itself has a good string sound, while bandpass filtering is easily added to a sawtooth to get a horn or reed output.

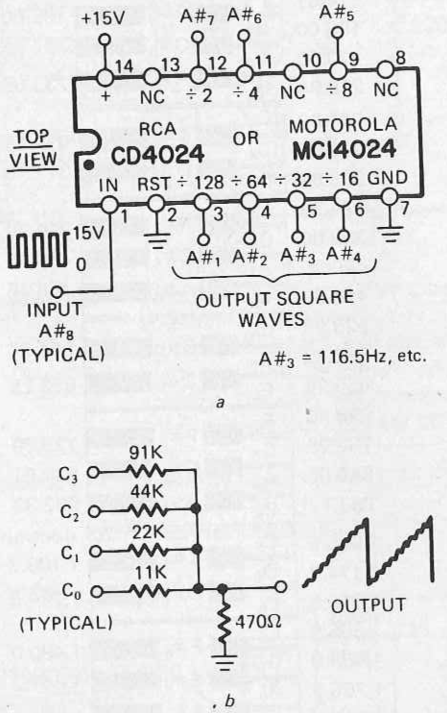


FIG. 2—DIVIDE BY 128 provides lower octaves for any note. a—Circuit for square waves. b—Adding resistors for sawtooth.

A tempo generator

How do you get a stable, cheap, wide-range square wave oscillator that's good enough as a monophonic note generator, but also is useful for rhythm and clocking, and easily drives TTL and CMOS to boot? With the *Signetics* 555 or *Motorola* MC1555 of course. This $1 IC can't be beat as a stable oscillator. Figure 3-a shows details. You can vary the resistance from 1K to 3.3 megohms, and make the capacitance anything you want above 500 pF or so. Output is usually a rectangular wave. If you need a square wave, make R2 much bigger than R1 or else add a binary divider to square it up. Figure 3-b shows how you can build a triggerable monostable or pulse generator out of the same circuit. This is useful for synthesizer envelope generation.

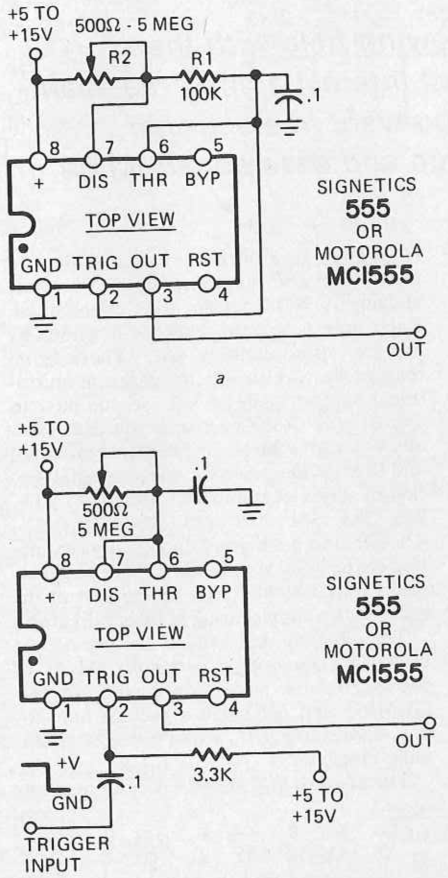


FIG. 3—USING THE 555. a—Astable, or rectangular wave generator. b—Monostable or pulse width generator.

A voltage-controlled oscillator

Many synthesizer systems are based on *voltage controlled oscillators*. Apply an input or control voltage, and you get an output frequency which you use as a tone source. Music VCO's have to be very stable to be useful. They also have to have a wide range. Ideally, they should respond in a log manner, but a log converter is more often added to the input of the VCO as a separate circuit. VCO's also should be able to put out a good looking sinewave for flute-like tones, as well as a square or triangle output. The *Intersil* 8038 does the whole job for under $3. A "baseline" circuit is shown in Fig. 4 that should get you started. Control voltage ranges from the positive supply to three volts or so *less*. The sinewave can be adjusted to below 0.5% distortion easily.

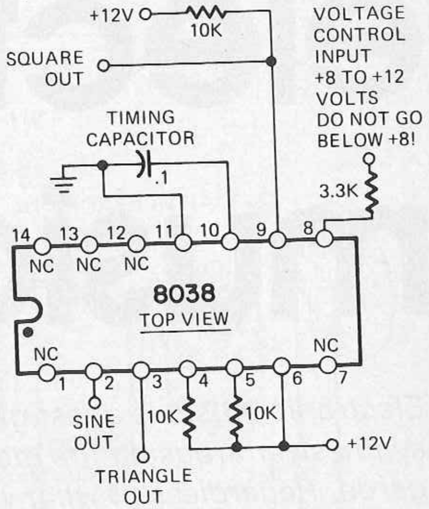


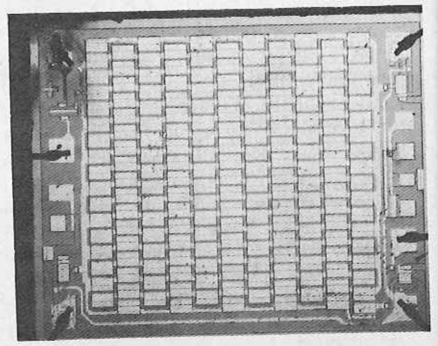
FIG. 4—SIMPLE VCO using Intersil 8038.

A dual operational amplifier

A good "741" style op-amp is essential for any electronic music circuit. An operational amplifier does at least three good things for you—it gives you controllable gain; it eliminates interaction and coupling between multiple inputs; and it gives you a versatile system gain block for active bandpass filters and things like this.

The *Motorola* MC1458 and the *Signetics* 5558 are typical dual "741" type circuits. Cost is around a dollar. The 5558 is in an easy-to-use 8-pin minidip package.

Figure 5-a shows the voltage follower connection. It gives you unity gain, a very high input impedance, a low output impedance and does not invert the signal. Figure 5-b is a voltage follower with gain. Figure 5-c shows the inverting amplifier and mixer. The gain of each input is the ratio of its own input resistor to the feedback resistor. The input impedance equals the input resistor for any input, and the summing point may be considered to be a *virtual ground*. There is no interaction between inputs or crosstalk problems possible in this circuit; further, you can *scale* or individually adjust each and every input to its own signal level independently, while the feedback resistor can be varied as a master gain control. Figure 5-d shows a good, high-Q bandpass filter circuit you can use to independently control the Q (to 500!), the gain, and the center frequency on. Use this for formant voicing circuits, sinewave recovery, and anywhere else you might like to emphasize a narrow frequency band.



ITT TCA350 ANALOG SHIFT REGISTER is 185-stage bucket-brigade af delay line.

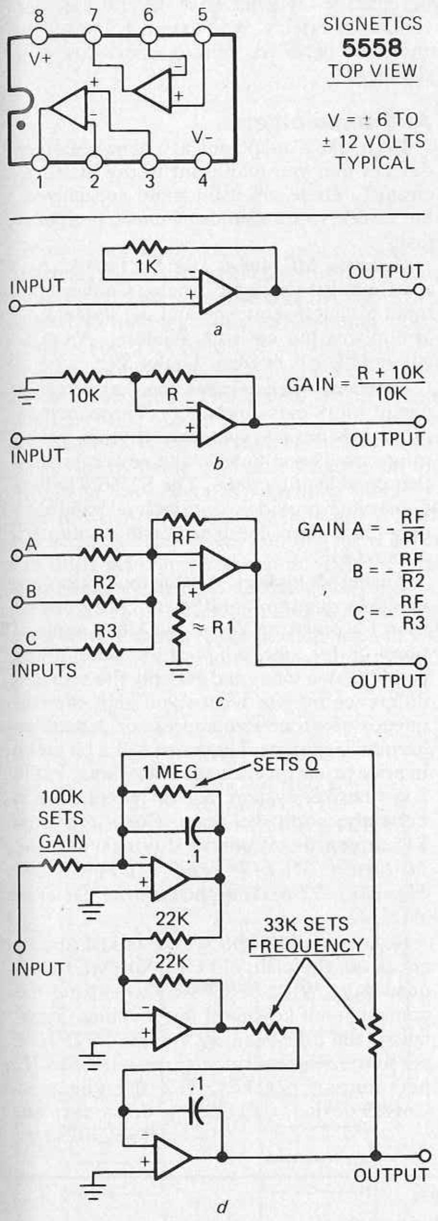


FIG. 5—A DUAL 741 STYLE OP AMP. a—voltage follower (high input Z, noninverting). b—Voltage follower with gain (high input Z, non inverting). c—Current summer or mixer −input is at virtual ground, no crosstalk is possible). d—High-Q bandpass filter.

By the way, if you are working at high frequency and high gain, the 741 style devices might not have enough bandwidth to do the job. If you need only a little bit more, try the Motorola MC1741S; for a whole bunch more bandwidth, go to the more expensive *National* LM318.

Six keyers at once

At the very least, music notes must be smoothly turned off and on without any key clicks or thumps. It's even better to be able to instantly vary the gain of the note so you can have complete control of attack, fallback, sustain, decay, snubbing, and perhaps even an echo. To do this, you need something that will behave as an electrically variable resistor. The circuit is called a *keyer*, an *analog gate*, or a *voltage-controlled amplifier*.

There are lots of bad ways to do this job. What you have is some circuit that is essentially *transparent* to the notes fed through it—it simply varies gain and nothing more. You must control the gain smoothly and do so equally well on the positive and negative portions of the envelope. Above all, you cannot let any portion of the envelope or control signal appear as an output, for this gives you a loud thumping.

Diodes have traditionally been used in organ circuits, but they thump, introduce distortion, and have a limited dynamic range.

An obvious choice is an integrated circuit four-quadrant multiplier—see below—, but these are far too expensive to use dozens at a time. Another possibility is an electronically controlled gain block such as the *Motorola* MFC6040, but it has too much gain for many applications.

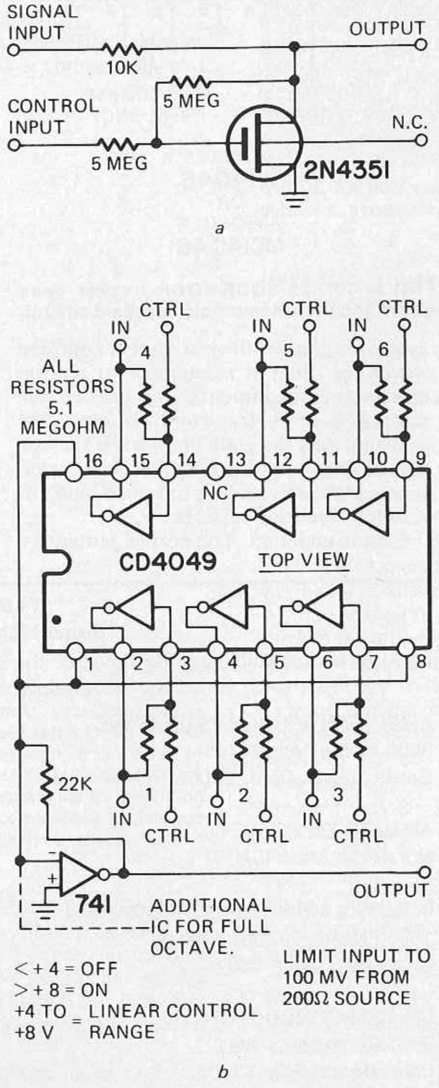


FIG. 6—HIGH-QUALITY HIGH-PERFORMANCE hex keyer or VCA costs only 30¢ per note. a—n-channel transistor as electrically variable gain control or keyer. b—CD4049 converted to six n-channel transistors.

The simplest *good* envelope keyer you can use is a single N-channel MOS transistor with some drain-to-gate feedback resistance. Figure 6-a shows the circuit. This gives you a linearly variable resistor, electronically controllable, and smoothly handles up to 10 volts of analog signal in either direction if the substrate lead is floated. Control voltage ranges from 4 or less for full off, up to 8 or more for full on; in between you get a good linear control range. For envelope and audio frequencies, there is absolutely zero feedthrough of the control signal. As a bonus, the control input looks into a high impedance that lets you use a small capacitor for the decay portion of the cycle. One typical discrete device is the *Motorola* 2N4351. At $2 or so, the cost is far cheaper than a multiplier, but still a bit steep if you use 97 of them at once.

Once again, it's digital CMOS logic to the rescue. A very few CMOS IC's can have their supply shorted to ground and thus disabling all the P-type transistors in the package. This leaves you with a block of N-channel MOS transistors that are ideal for gain control. You can get two and possibly three in the CD4007 (RCA) or the MC14007 (Motorola) devices in a dollar package. Most of the other devices, particularly the CD4009 and CD4010, have protective diodes arranged in such a way that you can't do this. The diodes are differently arranged in a new device—CD4049 (RCA) and MC14049 (Motorola). With this package, you get *six* voltage controlled amplifiers in a single integrated circuit. Cost now is around 35c per amplifier, but this will drop to around 10c per amplifer shortly.

Figure 6-b shows a full proportional control system with complete, thump free, control of attack, sustain, and decay.

Note that in both circuits, the package *ground* and *positive* terminals are tied *together* and form the *output*. The traditional inverter "outputs" are the signal or timbre input and the traditional inverter "inputs" receive the envelope commands. 3 volts or less is off; above 6 volts is [illegible] in-between you get a smooth control range. Best input signals are less than 100 millivolts high and from a 300-ohm or less source impedance. The op-amp builds this back up to a volt or two output, eliminates crosstalk, and prevents negative feedback from reaching the gate circuit.

An analog quad switch and sample-hold

While you are looking at CMOS, check out the RCA CD4016 or the Motorola MC14016. Either of these has four separate analog off-on switches that you can apply up to ten volts of peak-to-peak signal to. You use the same circuit digitally, frontwards as a one line to four line distributor, backwards as a four line to one line selector, or as four separate switches. Unlike virtually all other IC logic families, the signals can go through the switch in *either* direction.

Figure 7-a shows the IC. Figure 7-b is a digital or analog one-to-four distributor. Figure 7-c is a digital or analog four-to-one selector. Finally Fig. 7-d shows how you can build a *sample-hold* amplifier with one quarter of this package, a good Mylar capacitor, and an operational amplifier. Sample-holds are useful in synthesizers for remembering what frequency a note was after a key is released so the decay cycle can

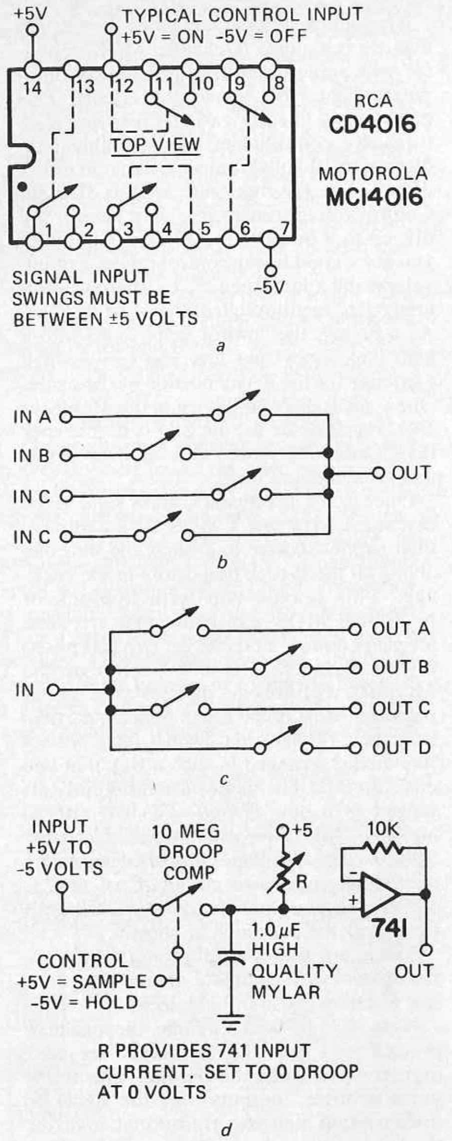


FIG. 7—A DIGITAL OR ANALOG quad switch. a—Circuit. b—Data selector—analog or digital. c—Data distributor—analog or digital. d—Low cost sample-hold.

be completed. The 4016 costs around $1.50. A complete sample-hold can be built for less than a dollar, since you need ¼ of this package, ½ of a dual op-amp and a capacitor. As with other CMOS circuits, the input control signal works into an open circuit. −5V is OFF; +5V is ON.

A tracker or gliding VCO

One of the biggest problems in any synthesizer is doing glides, sweeps, and trombone effects on a keyboard instrument. A circuit originally used by Olsen in the pioneer RCA synthesizer work to do this was called a *glider*. Today, it's called a phase-lock-loop tracker, and it is available as the RCA CD4046 or Motorola MC14046.

What the circuit does is this. You send it a frequency. It grabs onto that frequency from the one it is already at. You can control *how fast* the grabbing takes place. It can be nearly instantaneous, or it can provide a glide or sweep.

The circuit has an internal oscillator that compares its frequency against an input and then provides an error correction signal. You add a capacitor to slow down the response time to ''errors''.

Now, there're bunches of phase-lock loops available and you probably have already tried a few. The hangup here is that the IC you use must have at least a 1000:1 voltage controlled frequency range and must *NOT* be harmonic sensitive. This leaves out everybody but the MC4046. (The usual ''565'' type of PLL has only a 3:1 frequency range and is harmonic sensitive.)

One experimental circuit is shown in Fig. 8. Your input frequency can be a sine, square, or triangle or sawtooth wave. If you are trying to follow a more complex waveform, filter it *thoroughly* to recover mostly the fundamental, or use a comparator circuit for conditioning. The capacitor sets the glide time, while the bottom resistor sets the *damping* or the overshoot. Make this resistor too *small*, and you get wild ''Bounce'' effects.

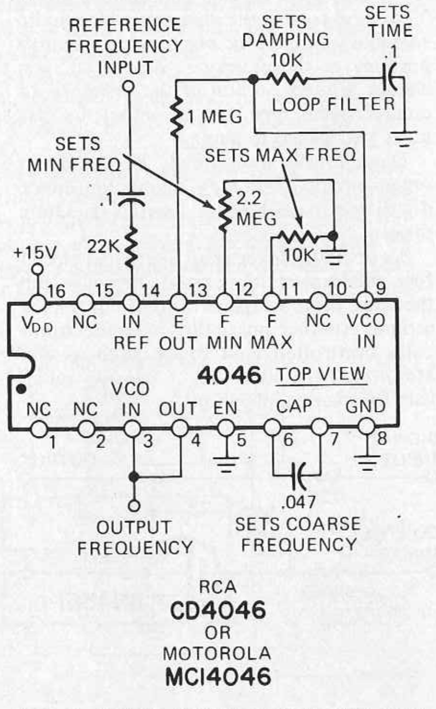


FIG. 8—PHASE-LOCK-LOOP tracker does glides and portamento from keyboard control.

Cost is under $5. The normal output is a square wave, but you can easily break the loop and put in a binary divider and sawteeth resistors. You can also divide the input as well, perhaps to follow a fifth above or below, an octave above, and so on. The potential is fantastic. Use several together for chorus effects. Add external ''noise'' to the error signal for vibrato, chorus, or randomness.

And some others . . .

Let's take a quick look at a bunch of other devices that you may want to use in music circuits. These are a bit more specialized, but can solve some unusual music problems fast:

Motorola MC1408-6 and MC1408-8. Six and Eight-Bit Digital to Analog Converters. Input a digital sequence and get notes out. Multiplying but not truly bilateral. An output amplifier is needed. Under $5.

American Microsystems Inc. has a whole line of MOS music products. These include older top octave systems, rhythm generators, rhythm counters, and newer devices that combine functions. The S2566 Rhythm Generator provides a complete bandbox-on-a-chip when combined with a counter. Around $18.

Analog Multipliers. Analog multipliers are true four quadrant multipliers. They can be used for precision keyer and VCA applications or for *ring modulators*, where you combine two tones and get only the sum and difference out, or where you shift the frequency of a tone to compress or expand its harmonic spectra. These are still a bit steep in price to use on each and every note, but in a synthesizer system, one or two of them is certainly worth the price. Costs run from $15 upwards. Typical devices are the *Motorola MC1494* and *MC1495*, the *Signetics* 5596, and the *Analog Devices* AD532J.

Besides the CMOS we've talked about, check out the plain old CD4001 (MC14001) quad gate. What better way to expand the contacts on a keyboard for coupling, translation, and transposition. It takes three IC's per new contact per octave, or ¼th of an IC per contact per key. It's the cheapest CMOS device, well under a dollar surplus.

—by Don Lancaster

TABLE 1
Some Manufacturers

(Be sure and specify *specific* devices; the majority of these circuits were designed for *non-music* applications.)

AMERICAN MICROSYSTEMS INC.
3800 Homestead Road
Santa, Clara, Calif. 95051

ANALOG DEVICES
Norwood, Mass. 02062

INTERSIL MEMORY CORPORATION
10900 North Tantau Avenue
Cupertine, Calif. 95014

ITT SEMICONDUCTOR
3301 Electronics Way
Palm Beach, Fla. 33407

MOSTEK INC.
1215 West Crosby Road
Carrollton, Tex. 75006

MOTOROLA SEMICONDUCTOR
Box 20912
Phoenix, Ariz. 85036

NATIONAL SEMICONDUCTOR
2900 Semiconductor Drive
Santa Clara, Calif. 95051

RCA SOLID STATE
Box 3200
Somerville, N.J. 08876

SIGNETICS
811 East Arques Avenue
Sunnyvale, Calif. 94086

TEXAS INSTRUMENTS
PO Box 5012
Dallas, Tex. 75222

what is a ROM?

ROM's have a fantastic number of uses and are widely available as you-build-it and factory-builds-it types. Here's what ROM's are and what they are good for.

HOW WOULD YOU LIKE TO BUILD YOUR OWN INTEGRATED CIRCUIT, perhaps to do a job you can't find some catalog item for? This used to cost $15,000 or so and take months of work. Today you can do it for $5 in minutes, with surplus units, and under $20 with first-run parts. The trick is to use an extremely versatile integrated circuit called a Read Only Memory or ROM for short. Let's take a closer look at this exciting integrated circuit and see what it is and how you can use it.

Actually, it would be much better if a ROM were called something else, for its name implies it's only good for computers. Worse yet, its name says there is something ''wrong'' or incomplete with the device. It would be best to call a ROM a ''universal code, state, logic, or sequence converter'', for this name at least hints at the thousands of different things you can do with the same basic device, custom-programmed to do a specific job. Since ROM is easier to say than ''UCSLOSC'', we'll go along with the original name.

Figure 1 shows the important parts of a ROM. There are a number of *input* lines, a series of *output* lines, some power connections, and an *enable* control that optionally lets you turn the outputs on or off or combine them with other ROM outputs in other packages. As you've probably suspected, a ROM is a *digital* device, meaning it accepts yesses and no's or **1**'s and **0**'s or positive voltage and ground as two-state input signals. It provides similar **1**s and **0**'s as two-state output levels. ROM's are available in most every logic family, including TTL, PMOS, CMOS, and ECL.

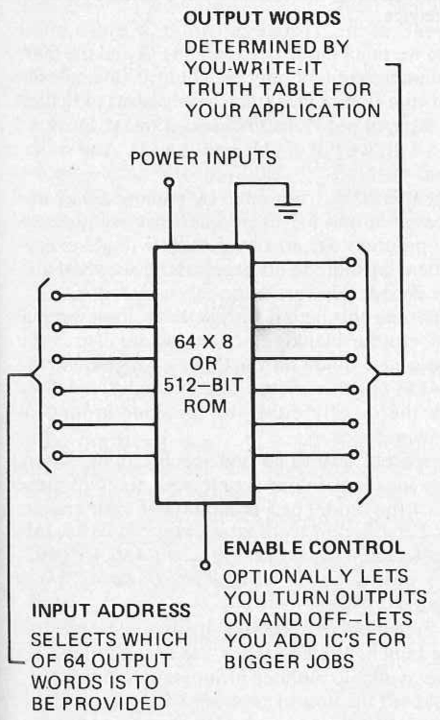
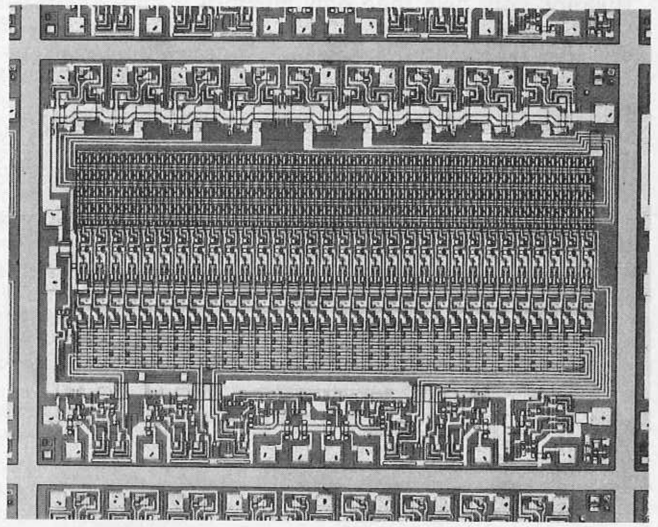


FIG. 1—ESSENTIAL PARTS of a typical ROM. This example is medium-sized and stores 512 bits of custom-programmed decisions by providing 64 possible 8-bit words at the outputs.

For each and every unique combination of input ones and zeros, a code word appears on the outputs. What this code word is or what it does in the rest of the circuit is *yours* to decide, for you can *teach* a ROM to do any one specific job for you.

For instance, if a ROM has six input lines, there are 64 (2^6) possible combinations of ones and zeros on the inputs, ranging from 000000, 000001, 000010, through 111110 and 111111. For each of these possible 64 conditions, you can select *any* output word you like, its maximum length determined by the number of available output leads. If you have eight output leads, each of the 64 words you select can be up to 8 bits long. Since we have 64 possible 8-bit words, we apparently have an internal ROM ''decision'' or ''training'' capability of 512 (8×64) *bits*. Each of the 512 locations can be a one or a zero per your choice, so there are apparently 2^{512} or *billions upon billions* of *different* things you can teach one IC to do.



INSIDE AN ROM you'll find a labyrinth of individual memory cells. Remember, actual size of this assembly is about .02 inch wide.

The arrangement of the ones and zeros you want is usually shown on a *truth table*, a state-by-state listing of all possible input combinations and the desired outputs you want.

How is this teaching done? There are several basic ways. If you need a lot of identical ROM's and are sure of what you want, you use a *mask-programmable* ROM. Here a final metal overlay connection pattern is set up for your particular program. All the ROM's made are identical up to this step. Your mask then customizes your order to the particular truth table you need.

More popular is the *field-programmable* ROM. You use these if you only need one or two, or aren't sure if your truth table will work, or suspect you will have to change things later. Some field-programmable ROM's arrive from the factory with a fuse at *each* possible location in the memory. Before you use the ROM, you go through a *programming* procedure that selectively blows out the fuses you don't want, leaving you with a custom pattern of ones and zeros that matches your truth table. You do the programming one bit at a time, usually applying a current of several hundred mA at a programming input. The current is increased till the fuse opens, and you then go on to the next fuse you want to open.

All this really takes is a variable power supply with a meter, but the ''zero defects'' nature of this work and its ''up the wall'' aspects make programming services very desirable.

Many electronic distributors offer nominal or free programming services and guarantee the results—provided, of course, that you wrote the truth table down correctly! Programming machines are also available that ease the problem. These cost several hundred to several thousand dollars, but speed up the programming tremendously and eliminate many error possibilities.

Other field-programmable ROM's use buried charge (electret style) layers or silicon bridges instead of nichrome fuses, but the result is the same. The ROM fresh from the factory is either all ones or all zeros, and you do something—usually by applying an excessive voltage or current to remove or implant something at every memory location—that changes the ones to zeros or vice versa. You then end up with the truth table you want.

Once programmed, the majority of ROM's are *permanent*; hence the name *read only*. If you made a mistake, you throw the IC away and start on a new one. On the other hand, since the programming is mechanical, it's forever independent of supply power. Turn your ROM off for a year and reapply power—and the truth table is still inside. A few newer ROM's are erasable by removing part of the lid and applying intense ultraviolet light. These are expensive and not too common yet.

Building your own read only memory

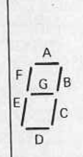
Let's build a "semi-discrete" ROM and see what it can show us about how the real ones work. Outside of doing it once as an exercise or to learn more about the process, going this route is complex and expensive compared to using the real thing.

Suppose we need a way of converting a 4-bit *hexadecimal* number into a 7-segment display so we can display the numbers 0, 1, 2, 3 . . . 9, A, B, C, D, E, and finally F with the letters handling states 10 through 15 and the numbers representing their own binary equivalents. A quick check of catalogs will turn up lots of different decoder/driver integrated circuits. This particular one seems to be rare, so let's pretend it doesn't exist at all. We have to use a ROM to build it.

Note that we'd go up the wall trying to build this out of simple gate packages—it would take a bunch of them and the design would take hours. With a ROM, the design only takes minutes, and a one-package solution almost always results.

We start by generating a *truth table* (Fig. 2). Our four input lines have 16 possible states (0000, 0001 . . . through 1111). We need seven output lines—one for each segment of the display. Let's provide eight to round things out and put a "count zero" detector on the eighth line. Each output line lights a display segment if it is positive and puts out a display segment if it is grounded.

INPUT				OUTPUT								PATTERN
D	C	B	A	A	B	C	D	E	F	G	H	
0	0	0	0	1	1	1	1	1	1	0	0	0
0	0	0	1	0	1	1	0	0	0	0	1	1
0	0	1	0	1	1	0	1	1	0	1	1	2
0	0	1	1	1	1	1	1	0	0	1	1	3
0	1	0	0	0	1	1	0	0	1	1	1	4
0	1	0	1	1	0	1	1	0	1	1	1	5
0	1	1	0	1	0	1	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	1	7
1	0	0	0	1	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	1	9
1	0	1	0	1	1	1	1	1	0	1	1	A
1	0	1	1	0	0	1	1	1	1	1	1	b
1	1	0	0	0	0	0	1	1	0	1	1	C
1	1	0	1	0	1	1	1	1	0	1	1	d
1	1	1	0	1	1	0	1	1	1	1	1	E
1	1	1	1	0	0	0	0	1	1	1	1	F



+ = *LIT* SEGMENT

FIG. 2—THE TRUTH TABLE WE NEED to build a 7-segment decoder/driver with a ROM. Its four input lines have 16 possible states. Each of its seven output lines drives one segment of a 7-segment display device.

For instance, we could connect our output lines to a MAN3 or MAN4 common-cathode LED readout. Positive current lights a segment. Voltage near ground puts it out. The segments are labeled A through G in the usual clockwise from the top manner.

To build the actual ROM, we use a bunch of diodes and a 74154 4-line-to-16-line decoder. This particular IC converts the four input lines into a one-down-out-of-sixteen pattern on our *intermediate output* lines, so that a 0000 input grounds the top intermediate output, a 0001 the next one down, and so on down to 1111 which grounds the bottom intermediate output line. Only one output line is grounded at a time; the rest remain positive. Figure 3 shows the circuitry.

Going to our truth table, 0000 should give us an output 0, lighting every segment but G, so we put a diode between line G and the 0000 decoded output. This diode conducts only on count 0000 and puts out this segment only at that time. On 0001, we only want to light B and C, so we evidently have to put zeros and diodes on A, D, E, F, and G. On 0010, to get a 2 lit, we put diodes on C and F. And so on, down the truth table.

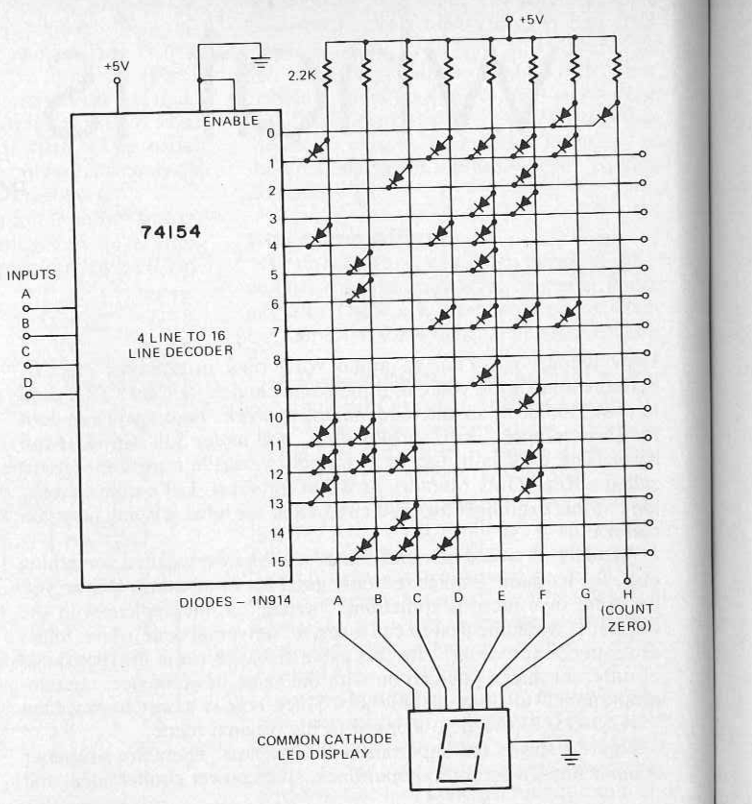


FIG. 3—OUR "FAKE" ROM is made from a 74154 4-line to 16-line decoder and a diode matrix to turn on and off the segments of a cold-cathode or LED 7-segment display device.

We mathematically generate the truth table by placing a 0 everywhere we want a segment out and a 1 everywhere we want a segment lit. We physically program our ROM by putting a diode everywhere we want a 0 and leaving a diode off everywhere we want a 1. And this completes our decoder/driver.

To get fancy, we can use the eighth output lines as a state 0 decoder that might be useful for blanking or somewhere else in the system. All this takes is a new diode on the 0 line. We can use the output enable on the 74154 to drive all the outputs high for a lamp test, and we can blank the display either by breaking ground or removing the supply power.

We used diodes to teach our ROM to do one specific thing. There are 128 possible memory locations in our simple ROM. Each of these locations can be given a 1 (no diode) or a 0 (diode) per your choice, so there are apparently 2^{128} *different* truth tables you can write. (My math book stops at 2^{101} = 2,535,301,200,456,458,802, 934,406,410,752. 2^{128} could be 134,317,728 *times* as large as this—you figure it out.)

Obviously we have a bunch of different truth tables we can write—A great heaping bunch. We can teach the ROM anything we like, consistent with the available number of inputs and outputs.

For instance, we could tell the ROM to subtract 3 from each input. Or multiply by 6.2. Or take the square root of it. Or we could tell it to decode and combine only certain states. We could make it play music. We can change codes or number systems. We can generate waveforms. There doesn't have to be any clear cut rhyme or reason relationship between inputs and outputs. If you can draw a truth table, the ROM will do the job for you—quickly and in a single package.

The *organization* of this particular ROM is called 16×8 or 16-8-bit words. Its potential memory locations are 128, so it is also called a 128 *bit* ROM.

ROM design is philosophically very different than older logic designs. The name of the game used to be a thing called "minimization", where you tried to get the logic equations in their simplest form and then build up a pile of gates to realize the "simplest" possible form. With ROM's you use *redundancy* instead. You take one logic block in one integrated circuit package with an incredible amount of redundancy—it can realize the "minimum" equivalent of *any* and *all* possible equations you could care to write consistent with the available inputs and outputs. You *ignore* the math and the simplifications! Instead, you just write down the truth table you want and program the ROM.

The benefits of redundant circuit design are overwhelming. The old way, you got a "minimum" logic design that took a dozen packages and took hours to design and debug. It was essentially unchangeable after design, particularly once it was locked into a PC board. The new way takes only seconds. Write your program, program the ROM and plug it in. The new way always works without any worry about glitches, races, disallowed conditions, sub-routines, and similar horrors. Changes? Simple. Just take out the old single IC that does the job and put a new one in its place. For everyday logic use, the textbook "minimization" techniques are an inexcusible waste of time and money once you get past a two- or three-package gate complexity. All they "minimize" is profits and the probability of success.

Commercial availability

Table I lists the commercial sources of programmable ROM's, one source of programmers, and one distributor that does programming. Table 2 lists a number of common ROM's and their organizations.

TABLE I

Some sources of ROM's and services:

CIRCUITS

Harris Semiconductor
Box 883
Melbourne, Florida, 32901

Motorola Semiconductor
Products
Box 20912
Phoenix, Arizona, 85036

Intel Corp.
3065 Bowers Avenue
Santa Clara, California, 95051

National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, California, 95051

Intersil, Inc.
10900 N. Tantau Avenue
Cupertino, California, 95014

Signetics
811 East Arques Avenue
Sunnyvale, California, 94086

Microsystems International
Box 3529 Station C
Ottawa, Canada

Solitron Devices
8808 Balboa Avenue
San Diego, California, 94086

Monolithic Memories, Inc.
1165 East Arques Avenue
Sunnyvale, California, 94086

Texas Instruments Inc.
Box 1443, Station 612
Houston, Texas, 77001

PROGRAMMING MACHINES

Spectrum Dynamics
2300 East Oakland Park Blvd.
Ft. Lauderdale, Florida

PROGRAMMING SERVICES

Semiconductor Specialists
Box 66125 OHare Airport
Chicago, Illinois, 60666

TABLE II

Here are a few currently popular programmable ROM's:

Part Number	Manu-facturer	Organi-zation	Bits
HROM-1-0512	Harris	64×8	512
HROM-1-1256	"	256×1	256
HROM-1-8256	"	32×8	256
HROM-1-1024	"	256×4	1024
HROM-1-2048	"	512×4	2048
IM5610	Intersil	32×8	256
IM5623	"	256×4	1024
MCM5003	Motorola	64×8	512
MCM5005	"	256×4	1024
MCM10139	"	32×8	256
MCM10149	"	256×4	1024
N8223	Signetics	32×8	256
N82S26	"	256×4	1024
SN74186	Texas Insts.	64×8	512
SN74188	"	32×8	256

One programmable ROM that's showing up quite a bit in the surplus market recently is the Signetics 8223. Costs have gone as low as $5 each. It is shown in Fig. 4. It's a bipolar device, DTL and TTL compatible and works with a single +5-volt supply. Operating speed is a fraction of a microsecond.

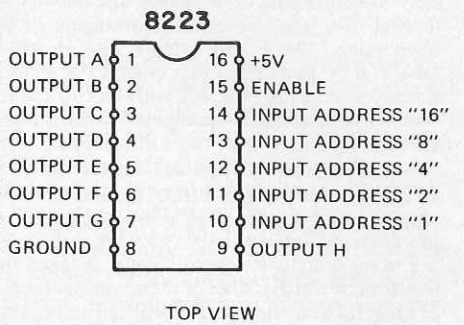


FIG. 4—A POPULAR PROGRAMMABLE ROM that is inexpensive, readily available, bipolar, and TTL compatible. Supply voltage is +8 V supply current is 65 mA. Connecting "enable" to ground provides an output code. Connection to +volts floats the output.

By the way—*preprogrammed* ROM's available surplus are only useful if you know *exactly* what they are and what they can do for you—a random or unknown program is totally worthless and essentially impossible to decode.

When do you use a ROM?

You use a ROM anytime you want a group of input numbers to be somehow related to a second group of output numbers, especially when you can't find a stock IC to do the job. ROM's become particularly attractive if the job seems hopelessly complex for construction using gate packages.

There are several different ways to use your input numbers. If you feed your ROM one number and then get a new one out on a random basis, you are using the system for *code conversion* or *table lookup*. If you sequentially go through your inputs, you have a *waveform generator*. Here the outputs provide an orderly progression of state changes, perhaps to generate a sinewave or a music note. If you use your inputs as separate logic inputs instead of feeding them a whole word at a time, you have a *programmable logic array*. Similarly, if you route your outputs to separate and distinct places, you have a *sequencer*, a *controller*, a timing generator, or a rhythm generator.

To really get fancy, you can let a ROM control *itself*. To do this, you store or latch the outputs each cycle and use the *last* output to provide the *next input* address. This way, the ROM marches through its truth table in a prescribed and controlled way. You use this for unusual length *counters* and computer *microprogramming*.

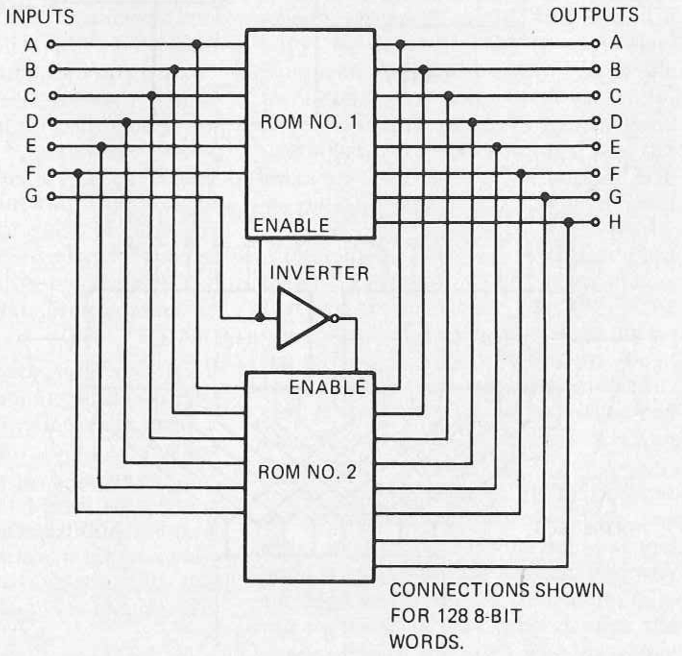


FIG. 5—HOW TO EXPAND ROM's by using several of them. Note that the doubled storage only offers one new input lead.

ROM design is easy. First, you make sure you really need one and that nothing is available commercially to do the job. Then you write your truth table. Then you find a ROM that fits it. Then you program the ROM.

If your truth table seems hopelessly large, you try to *minimize* it through several tricks of the trade. These include removing mirror images (such as generating only one quadrant of a sinewave), putting easy-to-realize functions *outside* the main ROM, using multiple trips through the ROM, factoring, rearranging to fit ROM organizations, eliminating "don't care" states, and so on. Virtually *every* truth table can be minimized in a complex system. If you have reduced things as far as possible and you still can't fit it in, you go to a larger ROM or several ROM's combined with input steering and output enables.

Note that you don't double the inputs when you add a second ROM to a first one—all you gain is one extra input. Since you only doubled the memory capacity, your addressing has only increased by one power of two. Seven lines have twice the storage capability of six. If you are using 6-input (64-word) ROM's, it takes *two* of them for seven inputs (128 words), *four* of them for eight inputs (256 words), *eight* of them for nine inputs, (512 words), and so on. Figure 5 shows how you combine ROM's with their enables.

There are several stock organizations of ROM's 16×8, 256×1, 64×8, and 128×4 being common smaller ones. Sometimes you can rearrange things with a latch or a data selector to change the organization if you want to. For instance, if your particular ROM has eight output leads, and you only need a 4-output word, you can use a 4-pole double-throw data selector (74157) to pick either the top or bottom four bits. This *doubles* the number of words you have available. On the other hand, you can provide two 8-bit latches on the output and enable them on *alternate* addresses. If you look at all 16 outputs at the right time, you get a 16-bit output word. Of course, to get this, you've cut the number of words in half and reduced the possible operating speed at the same time. Figure 6 shows how you can change the organization of a ROM to fit your needs.

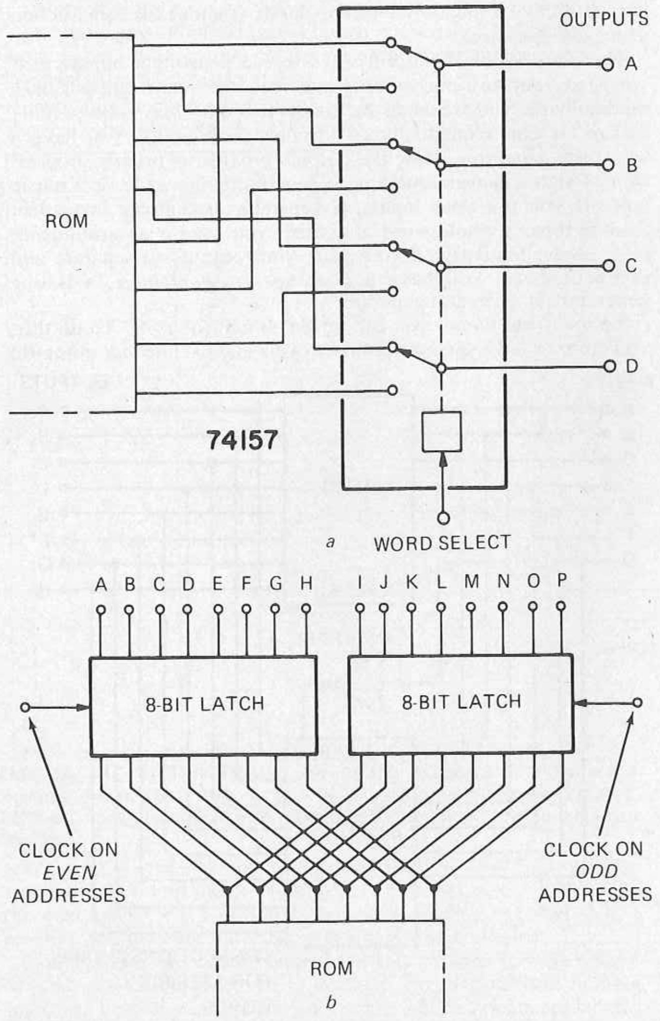


FIG. 6—THE DATA SELECTOR at (a) gives you twice as many output words of half the normal length. Using the setup at (b), the odd addresses are saved until even an address arrives. Output is half as many output words of twice the normal length.

Sometimes a special custom organization will help. This was done for the time-zone-converting ROM on the *Radio-Electronics* Superclock (July 1972), where a 384×6 MOS ROM was used with a simple external OR gate to convert 2400-hour time to any timezone in the world. "Non-binary" organizations normally cost quite a bit of money and are not available in field programmable units.

Even if you are planning on using a bunch of identical ROM's, you first use programmable ones, and then later go on to the cheaper mask-programmable versions. The breakeven point is typically several hundred identical units. A few dozen identical ROM's are easily copied or duplicated on a small programming machine. If you do go to mask-programmed units, ROM-PROM pairs make the changeover easy.

Stock read-only memories

Besides custom patterns, you can get stock pattern ROM's preprogrammed and ready to use. There is no masking charge for these, since they are a popular enough pattern that lots of them can be sold. Very common examples are the character generators such as we used in the *Radio-Electronics* TV Typewriter (September 73) and the TV Time Display (scheduled for a forthcoming issue).

There are several types of character generators. Most of them accept a 6-bit ASCII standard computer code on one set of inputs and some system timing on some remaining outputs. A *row* output character generator is designed to work with TV sets. It puts out a bunch of dots or *undots* on its output lines. These go to a TTL shift register and are then clocked out as video. A *column* output character generator works sideways and puts out a vertical group of dots and undots useful for moving message signs and strip printers. Either type costs around $12, but a bunch of support circuitry is needed.

Other stock ROM's include code converters, particularly to get from the specialized SELECTRIC and EBCDIC codes to ASCII and back again. Trig tables for sine and cosine generation are also fairly common, although still a bit steep in price. A vastly different stock ROM is the American Microsystems S2566 Rhythm Generator used to generate the accompaniment beats (waltz, tango, etc.) on an electronic organ.

In addition, many ordinary IC's are really ROM's in disguise, for the semiconductor people long ago found out that it's easier to design one redundant ROM pattern and then change the metallization overlay than to relay out and make separate IC's for each and every special function. The Motorola MC4000 series of TTL uses several ROM functions.

Applications

We've already seen that a ROM can be used anywhere you want to convert one group of digital words to a second group of words, either on a one-at-a-time, a sequential, or a let-the-first-one-decide-the-next-one basis. The wilder or the more unusual the relationship between the input and output, the better a ROM will work, for you work directly with the truth table. Competing systems require deriving all the individual logic relationships between input and output, and cannot normally be done in a single IC.

So far, we've talked about display decoder drivers, character generators, sinewave generators, electronic music, time-zone converters, and code converters. Let's take a quick look at some other possibilities.

Frequency synthesizers and digital programmers often use thumbwheel switches. The numbers of the switches indicate a channel number or a frequency, but the circuitry inside may take entirely different numbers to operate. Rather than use an expensive special switch, a ROM performs the internal conversion—the operator sees his number and the circuitry sees the number it needs at the same time.

Sinewaves are easy to generate by taking a counter and a sine-lookup ROM. Add a D/A converter for a low-distortion sinewave oscillator of constant amplitude that can go down to ultra-low frequencies without any large parts.

ROM's are used in cathode ray tube display systems for pincushion correction, dynamic focus and convergence, and so on. Besides generating dot-matrix characters, ROM's can store and generate whole messages as well. Often you generate the fixed portions of a message in a ROM and add the changing part to it. You can also scramble or unscramble data with ROM's, throwing away what you don't want and rearranging things to get a needed format.

Any logic equation you can write in truth-table form is also easily handled by ROM's. The one-package solution and instant design are top advantages. Besides, the circuit is trivially easy to change—you simply replace the ROM. Compare this with a traditional "minimum" logic design of several dozen packages and locked-in interconnections.

Clocked logic where the ROM output sets the next input address offer all sorts of possibilities for unique counter sequences and minicomputer microprogramming. You normally initialize your sequence with a latch reset or the enable. Loops are done by returning to the same address and branching is handled by creative use of a new input, or an Exclusive ORing change of an output word.

In fact, *anywhere* you want to change one set of numbers into another set, or anywhere you want to change what the signals on one bunch of leads are doing, you can use a ROM. And at last, the prices and availability are good enough that you can seriously consider using your own custom IC as a routine solution to a wide variety of digital problems.

SIMPLE SCOPE SERVICING

by JACK DARR
SERVICE EDITOR

THE CATHODE-RAY OSCILLOSCOPE IS THE fastest, most accurate, and *simplest* test instrument in a TV shop! We've been telling you guys that for too many years now. There are still entirely too many of you who persist in thinking of it as a very complicated, hard to use, scientific instrument. It is *not*! It's the best, fastest and simplest instrument in the place for giving you the answers to questions that can't be answered in any other way. The most common of these, and the most useful, is, "Is it there or isn't it?"

Typical instance: The set wiggles, bends, rolls and acts up, along with other symptoms. This kind of multiple symptom thing generally means that you have a feedback loop somewhere, causing cancellation of sync, assorted oscillations, and other things. How to find out? Pick up the scope probe and jab it on each of the *filter capacitors*, one at a time.

What should you see on each of these test points? *Absolutely NOTHING*. Nice straight line on scope-screen, indicating "pure dc". The dc power supply should have zero impendance to ground. If it doesn't, you'll have a beautiful feedback path through the dc power supply, which is the only thing common to *every* stage in the set. So if you see any "signal" when you scope the filter capacitors, something is darn well wrong. It almost has to be some fault in a filter capacitor; either open or low in capacitance. They are supposed to take out *all* signal from the dc power supply lines.

Double-check; leave the scope on the capacitor and bridge a good one across it. If the signals disapper and the symptoms clear up, there you are. By actual time-tests, this can be done in less than *30 seconds*. How fast can you get?

What do we mean "signal" in this test? *ANYTHING*. Set the scope for a reasonable vertical gain (the heater supply of your tube-tester is a handy source for rough voltage calibration. Voltage rms times 2.8 gives you the peak-to-peak reading. Never mind the horizontal sweep frequency! Jab the probe to the filter terminals, and if you see *ANY* vertical deflection at all you've got trouble! If you insist on fooling around, you can adjust the sweep and find out what frequency it is, but this is immaterial. Anything you see there is "ac" and means trouble.

More: this time you've got a dead amplifier. What kind? Any kind; rf, i.f., video, color. You know that signal is going into it, but it isn't coming out. How do you know you've got a signal going in? Because you can *feed* it in or rely on a known air signal. This can be checked with the scope, too. Example; hit the video detector output of a TV set. If you see about 1 to 2 volts p-p video at this point, but there's no picture on the screen, the video amplifier has a normal input but no output.

The "signal-path" in ALL amplifiers is a *series circuit*. If it is going in but not coming out, the path is broken somewhere. You must find out where. Start at the input and follow it. In a tube set, this would be grid-plate-grid-plate and so on. Transistors: base-collector-base-collector, etc. When you go through an amplifier stage, you'll see a voltage gain, in tube circuits. In transistors, you may not see much voltage gain, but you will have output. If you get out about as much as you put in, OK.

Somewhere along the line, you'll find a stage which has input but no output. There you are. On one occasion, I traced out an audio amplifier and located an open coupling capacitor between the second and third stages in a little less than 65 seconds! (Of course, I had the service data!) Even cold-turkey, you can usually follow the signal path, since you *can* identify the input and go from there.

In stereo amplifiers with one dead channel, you can tie the inputs together and "A-B" or cross-check between the same points in each channel. This will tell you exactly where you're losing the signal. It'll also tell you where distortion starts, if that's what you're looking for.

A lot of you have problems with color TV. Here again, the scope will give you fast, accurate answers that you can not get with any other test instrument. A series of "bang-bang" tests like those I just mentioned will check out a color bandpass amplifier, demodulator and color-difference amplifier stages quicker than you can say "Complementary symmetry integrated circuits". ((Speaking of that, the scope is the *only* way to go in any IC circuitry. If you have good signal in and no signal out of an IC, and dc voltage supply is normal, the chances are that the IC is bad!)

By feeding a color-bar signal into a color TV set, and tracing the unmistakable patterns through the bandpass, demods and diff-amps, you can identify any kind of color trouble. For loss of color, it will tell you where the color signals stop in the bandpass amplifier; if you see the characteristic "rocker" or Lazy-S pattern on the diff-amp grids, the demodulator *and* 3.58-MHz oscillator are working. If you see a flat-topped comb pattern on the differential-amplifier grid, the 3.58-MHz oscillator is dead. Instant identification of problem.

Let's kill another old superstition while we're at it. You do *not* have to have a 20-MHz. dual-trace, triggered-sweep scope, at about two grand a copy, to do this! They're very nice. But—you can make every one of the tests mentioned above with a narrow-band recurrent sweep scope in good working order! You won't see the exact waveforms, maybe, but you will get the *information* that you must have, from the Is it there or isn't it?" test!

One more and then I'll go. "Odd Color" problems. This often means that one of the bypass capacitors in the color stages is open, once again allowing a feedback loop to set up. Test: scope each bypass capacitor in the circuit. If yo see any *signal*, this is the bad one; that's what the bypass was put there for!

So; here you have an instrument which can do *more* for you than any other in the shop, by making your test and diagnosis time far shorter. So, why the heck don't you *use it*? I'm not talking to those of you who do use the scope, but to the guys who have one and leave it sitting in a corner gathering dust instead of gelt! Don't be afraid of it; it won't hurt you, and it won't hurt the sets; it *will* help you to diagnose any kind of problem in electronic equipment much faster and more accurately. So Use it! Use it! R-E