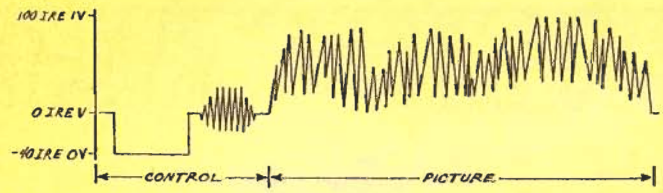
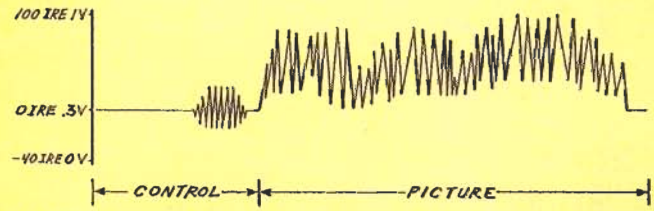
UNSCRAMBLING

DESCRAMBLING

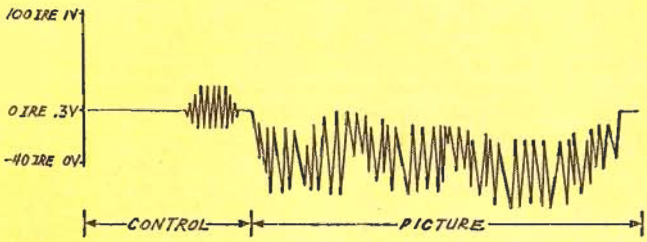
By Robert Grossblatt



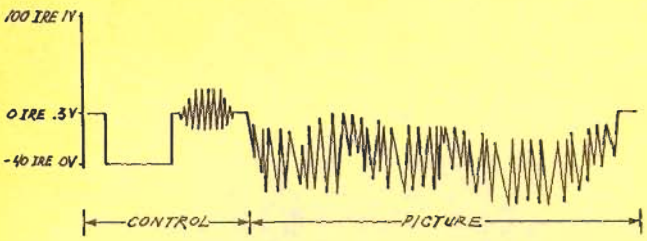
Typical line of video. Most of the line is devoted to the picture area, but it's the control area that we're interested in.



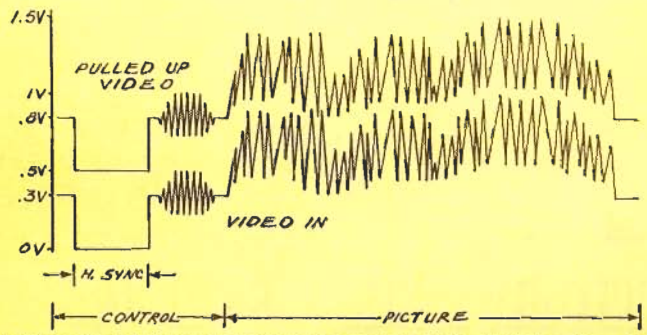
The SSAVI system can deliver video with suppressed horizontal sync and normal video.



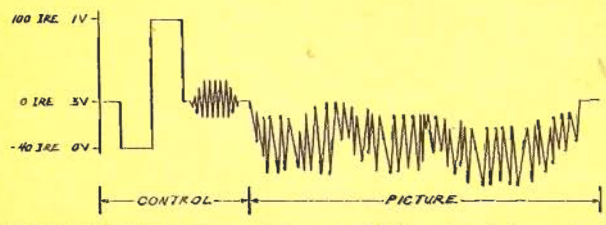
Suppressed horizontal sync and inverted video is also possible with the SSAVI system.



Here's what normal sync and suppressed video look like.



A video signal is normally 1-volt peak-to-peak, but after buffering, the relative voltage level of the signal is raised by 0.5 volts. Then, the only part of the pulled-up signal that falls below the TTL threshold of 0.8 volts is the horizontal sync signal.



This video waveform shows one variation on the SSAVI system. A change has been made to the 4.7-microsecond position normally occupied by the horizontal sync signal.

A GERNSBACK PUBLICATION

UNSCRAMBLING DESCRAMBLING

By Robert Grossblatt

I've been getting lots and lots of mail about video in general and scrambled video in particular. For some reason a lot of you really get enraged because some cable companies insist on scrambling certain premium channels. Before we go any further on this, let me tell you that I don't see anything wrong with it. Now, wait a minute—before you write me off as a stooge of the cable industry, let me finish.

The cable companies have every right to scramble whatever they want, although the rumors that some companies are scrambling everything they transmit—including the standard VHF channels—is going much, much too far. Premium stations and the pay-per-view shows are okay to scramble. What's not okay are some of the regulations that a lot of the cable companies insist are their God-given right to impose on you.

To begin with, hitting you with an extra fee for putting in another outlet is ridiculous. Some years ago the phone company did the same thing—anyone who added an extension phone on his own was risking life imprisonment or, even worse, being regarded as a not-nice person in the eyes of Ma Bell. That all went out the window years ago, and I think it's only a matter of time before the same thing happens in the cable-TV business. And, as far as I am concerned, the sooner the better.

The most annoying part of the cable system is the whole business of sending me scrambled signals, *and then telling me I can't do anything with them*! As I said, if the cable companies don't want me to get a particular channel (because I'm not paying for it, or some other perfectly legitimate reason), then don't send it to me. Trap it out of the line before the cable comes into my home. The additional cost of the traps has to be offset by the reduced cost of the cable box needed for the system, and the cost of installation should be the same because anyone with an opposable thumb and finger can put a trap on the line.

I agree that the signal coming into my home is the property of the cable company but, and this is important, at a certain time the real ownership of the signal becomes less clear. When the RF has been reduced to baseband video and has spent lots of milliseconds running around the inside of my TV set, I think things are a bit different and the cable companies' original claim of ownership is a lot weaker. And if I worked out a way to record scrambled signals and then descrambled them on playback, what then?

If I built a box that scrambled some of the channels currently sent to me in the clear, the cable company would look at me in a funny way, but I really doubt they'd care one way or the other.

Now that you know how I feel about this stuff, I'd like to show you how to descramble signals, but I

About Bob Grossblatt

Bob Grossblatt was raised in New York City and received a BS in Electrical Engineering from Rutgers University in the era of the sliderule. Although he planned on a career at IBM, he began working as an independent consultant shortly after graduating from college and has been doing that ever since. Most of his work in the electronics field has been in the conception and development of prototype devices for clients ranging from AT&T to NASA.

His career in electronics was interrupted at times by work in the movie business. He began writing about fifteen years ago and, in addition to the writing he has done for ELECTRONICS NOW, has also written several electronics books for Tab Books. Currently, his time is divided between circuit design, writing, and restoring old cars.

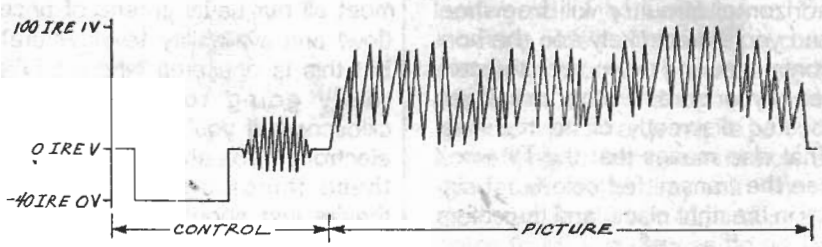


FIG. 1—TYPICAL LINE OF VIDEO. Most of the line is devoted to the picture area, but it's the control area that we're interested in.

can't because there are several ways that signals can be scrambled. It's sad but true that being able to descramble one system is no guarantee that you can descramble any other system.

The scrambling methods can be broken into two basic categories. The method you have in your home depends on the kind of cable service you have, how it's sent to you, and the economics of your viewing region. That last reason is important because the cable companies have to pay for the decoder boxes; the more sophisticated the way the signal is scrambled, the more the box costs. A cable company that has its franchise in a large city with lots of customers needs lots of boxes, and that translates into some serious numbers for the purchase of the boxes. And don't forget that the more extensive the scrambling method, the more expensive the equipment needed to scramble the signal in the first place.

Taking apart the video signal and turning it upside down and inside out is pretty simple, but putting it back together correctly is a different matter altogether. And the FCC keeps a careful watch on how close the reconstituted signal comes to real video. If the new signal is too messy, the FCC will give it a big thumbs down.

The most common approach to scrambling video involves manipulating the information in the horizontal interval. In the beginning, every cable company used the same method—they suppressed the horizontal sync pulse, which meant that the TV had no idea where each line of video started and ended. The sync pulse can be seen in the video waveform shown in Fig. 1. Without a sync pulse, the TV's horizontal circuitry will freewheel and you'll most likely see the horizontal weaving down near the center of your screen rather than being located discreetly off to the side. That also means that the TV won't see the transmitted colorburst signal in the right place, and the colors will be off as well.

To descramble the video, a decoding signal was sent, buried in the audio. It would restore the sync pulse at the proper time and for the proper interval. I don't want to spend a lot of time on this because there's as much chance of seeing this as there is of seeing a mastodon. Once upon a time they were everywhere, but they're long gone today.

The best way to get a good handle on the whole business of video scrambling is to get into the theory and the circuitry needed to turn the theory into practice. I'll assume that you understand the basics of a clear video signal as we go through the methods that are often used to mess it up.

Every scrambling method depends on altering some or all of the control pulses that are included in the definition of the standard video waveform. That means that the most basic operation of any scrambling/unscrambling system is the separation of the control information from the picture information. That isn't such a complex job because the NTSC standard was devised with a strictly mathematical timing relationship between every individual part of the signal. Therefore, looking at a video signal is somewhat like reading a street map—if you know exactly where you are, you automatically know where everything else is. Or, in the case of scrambled video, just where everything else is supposed to be.

Over the following pages we'll be looking at various scrambling methods commonly used by the cable companies. I'll go through the theory and show you how you can find out what your cable company is shipping to the back of your TV set. And yes, we'll be looking at the circuitry needed to descramble the signals—practical examples with component values.

I'll be using standard ICs that meet all our usual criteria of price (low) and availability (everywhere), but this is one area where you're really going to need an oscilloscope. If you're serious about electronics you should have one of these things anyway, because they're just about the most basic and essential piece of test equipment you can own.

How They Scramble

You don't have to be a rocket scientist to mess up video—that is true both aesthetically and scientifically. The hard part is to do it in such a way that you can put it back together again. This means that there has to be a rigorous approach to the task—almost a mathematical one—of tearing the video signal apart.

Take a look at—and get intimately familiar with—the typical line of video shown in Fig. 1. While most of the time on the line is devoted to the picture area, it's the control area where the real work is done. The video signal in the picture area determines what you'll be seeing on the screen but the stuff in the control area is what tells your TV where to put the picture and how it's supposed to appear.

The control area is blown up in Fig. 2, and the information in it is a graphed function of time and voltage. By the way, most video people like to talk about "units of video" rather than voltage for the same reason that audio people like to talk about decibels rather than voltage.

When the NTSC video standard was established, the two most basic decisions made were that it would range from 0 to 1 volt peak-to-peak, and that one voltage range would be reserved for picture and one would be reserved for control. As we go through our discussion on scrambled video, I'll talk sometimes about video in terms of IRE units and other times about voltage. The two are directly related as shown on the Y axis of Fig. 2.

The bottom line of the picture is 0 IRE units which is about 0.3 volts up the IRE scale. That point is important because it's both the defined level for black video (no picture on the screen) and the upper limit for any control signals. (There's a slight ambiguity here when you examine the colorburst.) For the moment, we can consider everything above 0.3 volts as picture and everything below that as non-picture.

That signal definition is the basis for most of the hardware in every NTSC-compatible TV ever made. Your TV contains circuitry that expects control information to be below 0.3 volts and picture information from 0.3 to 1 volt. That's important because it is the starting point for

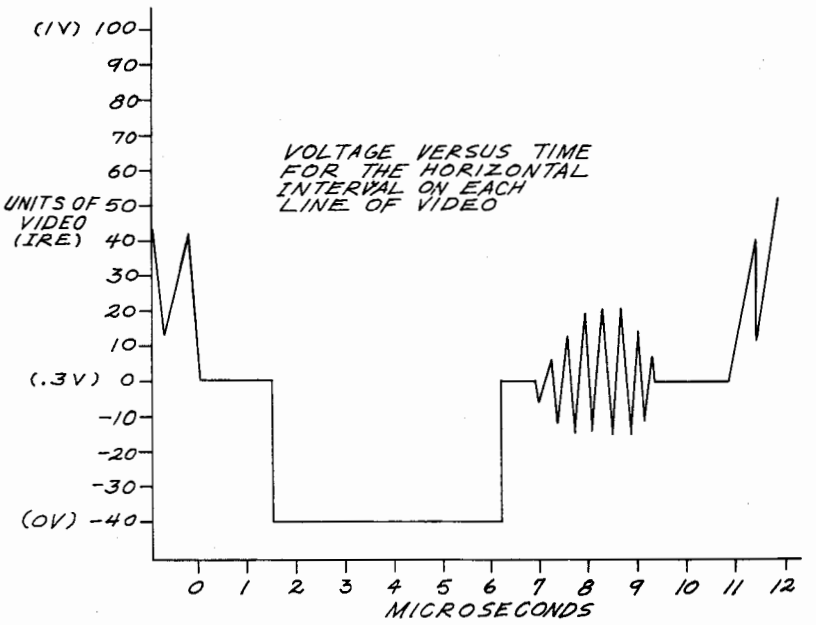


FIG. 2—THE CONTROL AREA. The NTSC video standard says that the signal can range fom 0 to 1 volt peak-to-peak.

scramblers; when you get rid of some of the control information, a standard TV can't display the picture. Remember that the horizontal sync pulse defines the end (or, depending on your point of view, the beginning) of a line of video. If the TV doesn't see it, it won't know how to display the line on the screen, and the result will be that the TV will end one line and start another one at some random point on the screen.

The freewheeling retrace frequency of the TV will come close to the one sent by the broadcaster, but it won't match exactly. What you'll see on the screen will be something like Fig. 3-*a*. The curved line running down the center of the screen is the horizontal interval sent by the broadcaster.

Three things are happening in Fig. 3-*a*. The first is that the line is curved because the horizontal circuitry in your TV runs at a frequency that's not exactly the same as the broadcast horizontal frequency. The TV can accept a certain amount of drift in the horizontal frequency. Once upon a time a horizontal control was built into on the TV so you could hand tune the TV to the received signal. Although that control isn't around any longer (except sometimes as a trimmer on a circuit board inside the TV), the tolerance is still there. Modern TVs can automatically lock onto the broadcast horizontal frequency so there's no reason for the horizontal control to be accessible.

The second thing that's happening is that the line is in the center of your screen. The reason for that is simple. The TV's horizontal circuit uses the received horizontal pulse as an instruction to move the beam back to the left side of the screen. Because the scrambled signal has anything but a recognizable horizontal sync pulse, the TV zips the line back to the left side of the screen whenever it reaches the right side. Because that has nothing to do with the signal it's receiving, the line usually shows up at some random spot on the screen. The TV's freewheeling frequency is close to the broadcast horizontal frequency, so the TV will start a new line at about the same point in the broadcast line. That means you'll see the broadcast horizontal interval on each line at more or less the same horizontal location on the screen. The result is a curved line down the screen.

The third thing happening on the TV screen is that the colors are messed up. Because the horizontal sync is missing, the TV circuitry isn't seeing the colorburst in the right place, so there's no reference for either the intensity or color of the picture. The TV then uses whatever it sees in the colorburst location as a reference for both the intensity and color of the image.

You can see now that by simply getting rid of horizontal sync, the resulting video signal will be completely messed up. The best way to appreciate that, and a good way to get into video hardware, is to build something to demonstrate how all this stuff really happens. That's right, our first piece of hardware is going to be something that will let you scramble video. And, as far as the law is concerned, I'm pretty sure that nobody's going to become very upset.

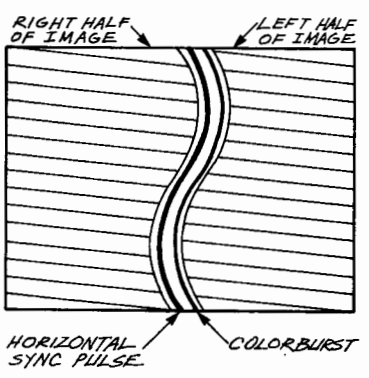


FIG. 3-*a*—A FREEWHEELING RETRACE won't match the frequency sent by the broadcaster. The curved line running down the center of the screen is the horizontal interval sent by the broadcaster.

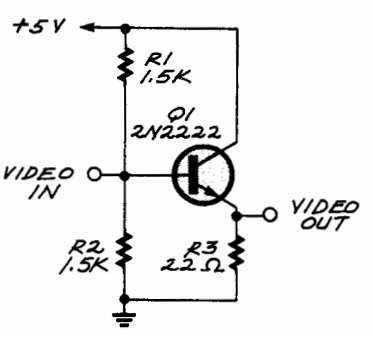


FIG. 3-*b*—A VIDEO BUFFER isolates one stage of hardware from another. The transistor is set up as a buffer and the level of the video can be controlled by the value of R2.

What we will need is a source of real live video.

That signal can be anything from an NTSC generator to a line-level video signal from the back of a VCR. You'll also need a scope to look at the video waveform and a TV to look at the picture. You can do without the latter but the former is a must. I'm not going to beat you up any more about getting a scope, but if you don't have one, get one.

To get started, because we're building circuitry that is going to use an external signal, the first thing we have to do is buffer it. That is done for two reasons. The first is that we have to be able to control the level seen by our video circuitry, and the second is so that a wiring error on the breadboard isn't going to send unpleasant voltages back to the signal generator or VCR. The results could be a bit nasty.

Video buffers are just like any other buffer—they're simple circuits that isolate one stage of hardware from another. Think of the circuit as being like an electronic fuse.

The easiest way to build a buffer is with a single transistor as shown in Fig. 3-*b*. The transistor is set up as a buffer, and the level of the video can be controlled by the value of R2. You can also put a potentiometer in series on the line feeding the video to the base of the transistor and trim the level that way.

Although the NTSC video standard calls for a signal that's 1-volt peak-to-peak, most VCR manufacturers don't strictly follow that standard when it comes to a video output signal. If you put the signal on a scope, you'll probably find that it's a bit higher than that. If that's the case, you should trim the level because the circuits we'll be building expect a 1-volt signal.

The only other thing to notice here—there just isn't much to the circuit at all—is that the video signal being fed to the base of the transistor is related to both positive voltage and ground through R1 and R2. The circuit is going to run on a regulated 5-volt supply; it must be steady because the level of the supply voltage is going to have an effect on the level of the video. Wire up the circuit show in Fig. 3-*b* and get the video source in place.

Side-tracking to SSAVI

The old suppressed-sync system was a one-way deal. If you got a box that could descramble one channel, it could descramble any channel. Which channels would be unscrambled was determined by one of the wafers on the channel selector dial. A position would be either jumped or open, which was a major cable company headache for two reasons. The first was that they had to open the boxes and solder or cut traces to configure the box for a given customer. The second was that some enterprising people realized what was going on, opened up their cable boxes, and reconfigured it themselves.

The only way the cable companies could guard against that was to use screws with oddball shaped heads to hold the box together. When that didn't work, they started using screws that had a left-hand thread. But enough history.

What the cable companies needed was a way to talk to each of the boxes individually, while they were in customer's homes. Making such addressable boxes also meant that several scrambling methods could be used; the boxes could be told which method was in use at any one time. Since that information could be sent to the box during the vertical blanking interval (while the beam was off the screen), the cable operator could change the scrambling method from field to field—up to sixty times a second. The boxes could also keep a serial number in an EPROM or some other storage device, which meant that boxes could be addressed individually and the descrambling circuitry could be turned on and off for separate channels from the main cable company office. The cable companies loved it.

Understanding that kind of stuff is a bit more difficult than the old suppressed-sync system, but if you take the pieces one at a time, it all gets cut down to manageable, bite-sized chunks. Although the cable company's scrambling delivery system became much more sophisticated, it was still faced with the same cost restrictions when it had to decide which of the available scrambling techniques to use.

One of the most popular choices was the so called SSAVI system. That's an acronym for Sync Suppression Active Video Inversion. It

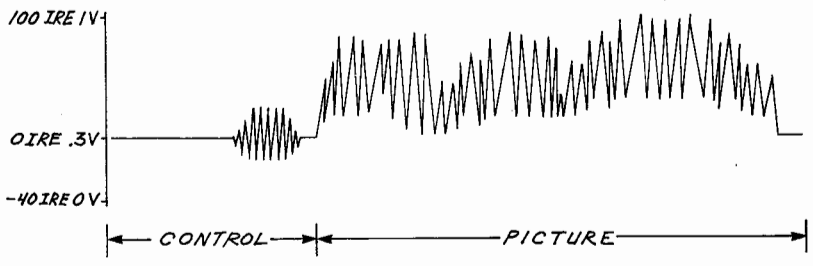


FIG. 4-*a*—THE SSAVI SYSTEM can deliver video with suppressed horizontal sync and normal video.

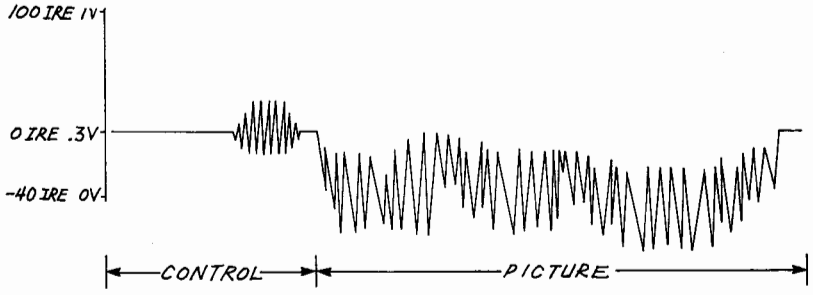


FIG. 4-*b*—SUPPRESSED HORIZONTAL SYNC and inverted video is also possible with the SSAVI system.

allows the video to be delivered to your doorstep in one of four flavors:

- Suppressed horizontal sync and normal video (Fig. 4-*a*).
- Suppressed horizontal sync and inverted video (Fig. 4-*b*).
- Normal sync and suppressed video (Fig. 5).
- Normal sync and normal video (we can forget this one).

Before we get into the nitty gritty of the SSAVI system, there are a few basic things you should know, because they tell you some interesting things about how the system works.

The first is that horizontal sync is never inverted—even if the picture is inverted. This means that any circuit designed to descramble it has to separate the two basic parts of the video line (control and picture first). We have to be able to turn the picture right side up (if needed) without inverting the control section as well.

The SSAVI system seems even more complex when you realize that the job of separating control and picture has to be done on lines that might very well have no horizontal sync pulse that can be used as a reference mark. In the older suppressed-sync system, the sync could be recovered from the gating signal that was buried in the audio; with the SSAVI system, there's nothing like that available.

The key to regenerating the video signal is based on the fact that all aspects of it are tied together in a strict mathematical relationship. If you can locate one part of the signal, you can determine where everything else has to be.

The broad picture for a descrambler, therefore, is to design a circuit that can identify one part of the signal, and then use the repetition of that signal as a reference for restoring the rest of the video. You should realize by now that we're talking about a phase-locked loop, or PLL. Even if the identifiable component of the video occurs only once a field (or even once a frame), that's still often enough to control the frequency of a voltage-controlled oscillator, or VCO, and lock the PLL to the received video.

This isn't as strange as it might seem. In a normal video signal, the reference for color is the colorburst signal that follows horizontal sync. The colorburst signal lasts only a bit longer than 2 microseconds, but it's used as a reference for the whole video line, which is about 63 microseconds long. As far as color correction is concerned, that means there's no real reference signal available for more than 95% of the line! The color phase for the rest of the line is based on the stand-alone 3.58-MHz generator that's a normal part of the TV set.

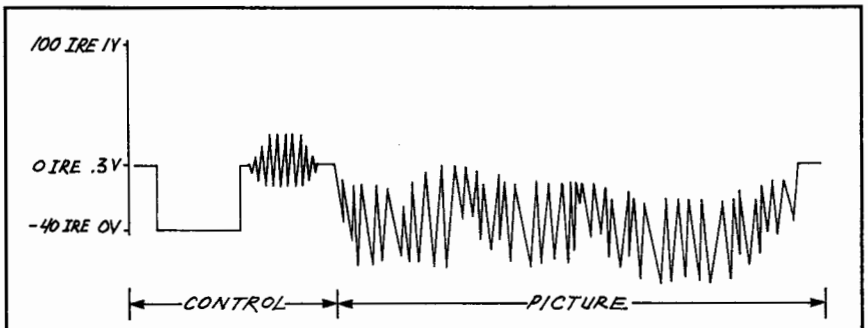


FIG. 5—HERE'S WHAT NORMAL SYNC and suppressed video look like.

Building a SSAVI descrambler isn't as easy as building one to take care of suppressed sync, but it's not as difficult as you might think. Before we start to work out the details of the circuitry, we have to draw up a comprehensive list of exactly what we want the circuit to do. A circuit designed to descramble the SSAVI system needs the following basic features:

- A means of knowing if the picture will be normal or inverted.
- The ability to generate horizontal sync pulses.
- A way to identify a definite point in the received video.
- A circuit to place horizontal sync pulses at the right point.

Some SSAVI systems also play games with the audio, but the methods used to hide the audio have been around for a long time. The audio is usually buried on a subcarrier that's related, in some mathematical way, to the IF component of the TV signal. We'll get into that briefly when we take care of restoring the picture.

The SSAVI system uses digital signals for security and access rights—the stuff that cable executives lie awake all night thinking about (instead of less-important things such as improving picture quality, increasing channel services, and widening the audio bandwidth). Because the first step in handling SSAVI scrambled signals is to locate a known point in the signal, we'll be using counters and other standard digital logic to keep track of where everything is supposed to be.

Build Your Own Scrambler

We haven't seen very much circuitry yet on our journey through videoland. That's to be expected, though, because video is a subject whose theory you should understand before you start building software. A video signal (shown in Fig. 6) is very complex, with many separate components that are mathematically related to one another.

If you look at a video signal on an oscilloscope, it will appear more or less like the lower waveform in Fig. 6. The most important component of the waveform is the horizontal sync pulse; if you do away with it, the TV won't have any reference for the beginning of a video line, and the image will be misaligned vertically.

Altering Horizontal Sync

Suppressing the horizontal sync is a simple, inexpensive, and relatively safe way to keep "unauthorized" viewers from receiving a coherent signal. So, to understand better how scrambling works, let's build a circuit that can alter the horizontal sync.

Because we're dealing with composite video, and we intend to play games with horizontal sync, the first thing we have to do is isolate the sync from the rest of the signal. That isn't very difficult—every TV in the universe can do it. Most modern

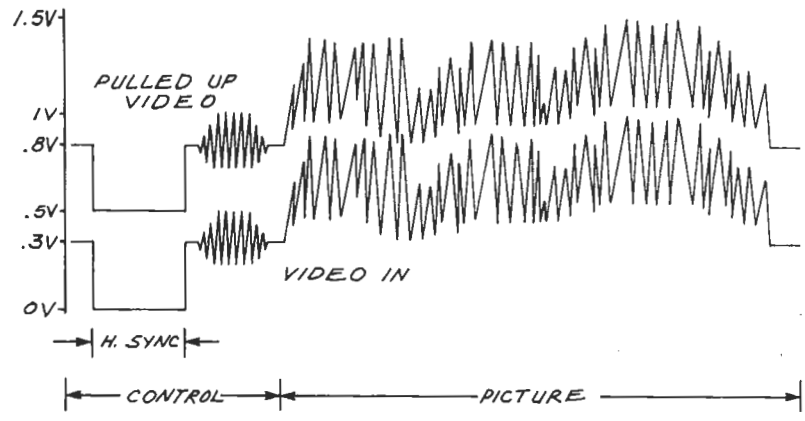


FIG. 6—A VIDEO SIGNAL is normally 1-volt peak-to-peak, but after buffering, the relative voltage level of the signal is raised by 0.5 volts. Then, the only part of the pulled-up signal that falls below the TTL threshold of 0.8 volts is the horizontal sync signal.

TVs either use a discreet sync separator chip or have the needed circuitry buried in the innards of some custom silicon. That makes things cheaper for TV manufacturers, but it's murder for people like us who have a hard time buying the chip in single quantities.

Fortunately, there's always more than one way to get the job done. In this case, it means looking at the voltage definitions inherent in the video signal, and seeing what we can do with them. Standard video has very strict voltage divisions; everything above 0.3 volts is picture information and everything below 0.3 volts has to be a control signal. (We haven't talked about vertical sync yet, but you'll find that the same voltage levels apply to it, too.)

When you have a 5-volt supply and a signal voltage with a 0.3-volt knee, you should immediately think about standard TTL logic. In that family, everything below 0.8 volts is low, which is exactly what we're looking for. That might not be immediately obvious, so let's go through it.

A video signal is 1-volt peak-to-peak but, by buffering it, the relative voltage level of the signal is raised by 0.5 volts. So, instead of ranging from 0 to 1 volt, the signal ranges from about 0.5 to 1.5 volts. The translated level of the control/picture voltage point is now about 1 volt (see the upper waveform in Fig. 6). You can see that the only part of the pulled-up video signal that falls below the TTL threshold of 0.8 volts is the horizontal sync signal.

The bottom line here is that we can build a sync separator from a standard TTL gate—in this case we'll use a 7486 exclusive-or (XOR) gate. All we have to do, as shown in Fig. 7, is feed the translated and buffered video from Q1 to one input of the gate, and tie the other input of the gate high.

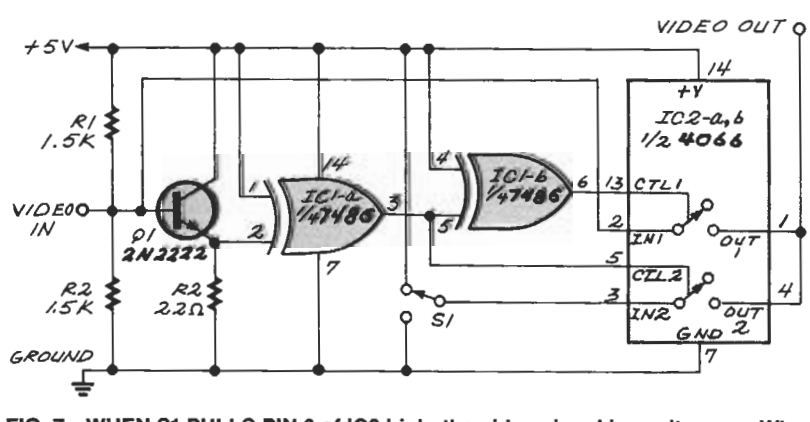


FIG. 7—WHEN S1 PULLS PIN 3 of IC2 high, the video signal loses its sync. When S1 pulls pin 3 low, sync is restored.

Suppression Circuit

If you work out the truth table for yourself, you'll see that the only time the output of the gate is high is during horizontal sync. The output at pin 3 of the 7486 is a TTL-level inverted version of the horizontal sync. That output is fed to another XOR gate, which inverts the signal and gives us a negative-going sync signal. Ability to provide both a positive and negative sync signal is the key attribute of the suppression circuit. We want to build a switch that passes video during the picture portion of the signal and be able to alter the signal during the horizontal sync period. That's what the rest of the circuit does.

The first part of the circuit is a picture/sync separator, and the last part is a picture/sync combiner—sort of. Even though we can put the sync back in, we also have the option of sticking in just about anything else we want in place of horizontal sync.

The combiner uses half of a 4066 analog switch as a double-pole, double-throw switch. (The analog switch contacts close when the control voltage is high.) The outputs of the switch (pins 1 and 4) are combined, but because the control lines of the switches (pins 13 and 5) are connected to mirror images of the horizontal sync signal, we can route the picture portion of the video signal to the switch output when sync is low (pin 6 of the 7486) and route horizontal sync to the switch output when sync is high (pin 3 of the 7486).

The single-pole, single-throw switch (S1) controls the input to pin 3 of the 4066. While it's neat to see the effect S1 has on the video signal when seen on an oscilloscope, this is one of those cases when you're better off seeing the effect on a TV.

Whenever S1 pulls pin 3 of the 4066 high (anything above the expected sync level), the video signal loses its sync and the picture on the TV goes totally haywire. If you've seen scrambled pictures before, you'll recognize it immediately. The

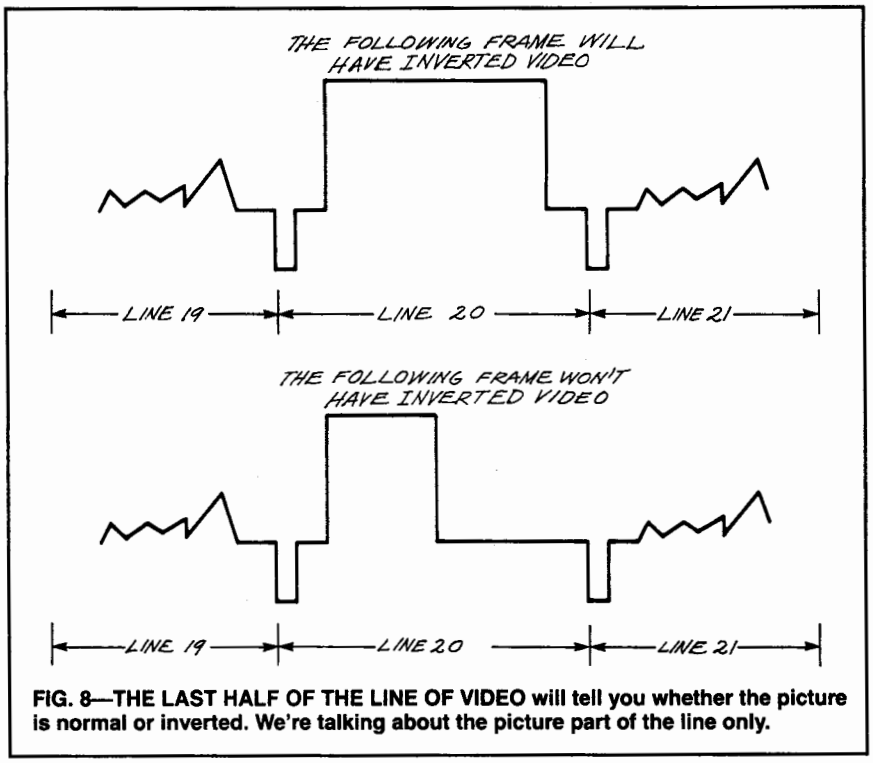


FIG. 8—THE LAST HALF OF THE LINE OF VIDEO will tell you whether the picture is normal or inverted. We're talking about the picture part of the line only.

left side of the picture will be on the right half of the screen, the right side of the picture will be on the left half. Down the middle of the screen will be the horizontal interval. When S1 pulls pin 3 low, sync is restored and so is the TV picture.

Putting It Together

We are not ready to go into the details of the scrambling business just yet, though. A successful scrambler not only has to take the video apart, but it also has to put it back together again. That is quite a bit more difficult. There has to be a way to encode the video signal so that the horizontal sync signal is restored at the right time, and for the right length of time. One outdated way that this can be accomplished is to bury the information in the 31.5-kHz audio subcarrier.

That's not so surprising when you realize that half that frequency is 15.75 kHz—exactly the same as the scan rate of the video lines on a standard color TV. There's not much point in going through all the gory details of recovering suppressed-sync video since it's about as useful as presenting a full tutorial on repairing telegraph lines.

Since suppressed-sync scrambling was figured out by signal pirates about five minutes after it appeared, the people in the television signal scrambling business moved on to more complex methods of screwing up the video signal. The most common method now in use combines a variation on the suppressed-sync method, inverting the video, and performing a lot of other weird stuff.

Back to our SSAVI

When the SSAVI system first started, there were some constants in the video signal that could be used to descramble it. Remember that the picture can be messed up in any one of three ways. (See page 5, column 1), and the instructions for the descrambler are transmitted somewhere in the vertical interval. The word "somewhere" is a late addition to the SSAVI system. When it first started, the descrambling information was always on the same line. That's where we'll start.

Once upon a time, the sanctity of the vertical interval was closely guarded by the FCC, but as alternatives to standard broadcast TV became more popular (cable, satellite, etc), more and more junk started to show up there.

When the SSAVI system started, lines 0 to 9 were left alone by a request of the FCC, but lines 10 to 13 were where the cable companies transmitted individual subscriber codes. Don't forget that there are unique ID numbers stored in an EPROM (or some other kind of memory) in the cable box. There's also logic circuits there to count the video lines, read the transmitted code, and match it up against the one stored in the box. These technical tricks are a big thing for the cable companies because it prevents a New York box from being used in California. The scrambling is the same, but the codes are completely different.

The decoder circuitry is also controlled by this coding process because a match between the transmitted bytes and the ones stored in the box will enable or disable the decoder. That is true for both the premium cable services and the pay-per-view events.

That kind of coding might be important to the cable companies, but it doesn't mean anything to us. We can build an experimental descrambler without paying any attention to them.

Since the video can be transmitted with either normal or inverted picture information, one of the tasks that has to be done by the descrambler is to tell the rest of the circuit what has been done to the picture. The place to find that information was originally in line 20, but it has been moved around since the system became popular. As you can see in Fig. 8, the last half of the line will tell you whether the picture is normal or inverted. Remember that we're talking about the picture part of the line only, and not the control section.

If the following field is normal, the last half of the line will be black, and if the picture is normal, the whole line is white. One of the things a decoder needs, therefore, is some way to detect the line and store the data it contains. The stored data is then used as a switch by the circuit to route the video through an inverter if the picture is being transmitted upside down.

This is pretty straightforward stuff. Since we're looking at only one piece of information, all we need is a place to store one bit of

information. Your basic piece of cake. The circuitry needed to detect the data, however, is a bit more complex. We need a reference in the signal. So establish a zero point for a line counter, and some counting circuitry to keep track of which line is being received.

You might be wondering what we can count if the signal is being scrambled. But remember, that in the vertical interval (the first 26 lines of video), the signal is being sent in the clear.

Now that we have an approach to handling the possibility of an inverted picture, the last problem to tackle is the one of varying horizontal sync pulses. Sometimes they're there, sometimes they're absent, and sometimes they're not at the proper level. Anything that unstable is a pretty poor choice for a reference signal. So, to avoid a mammoth circuit design problem, the best way to deal with it is to scrap the transmitted horizontal sync (even when it's there), and come up with a way to generate the signal ourselves.

That can also seem to be an insurmountable problem but, just as in the case of the inverted picture, the answer is going to be found in the vertical interval. Once again, remember that the first 26 lines of video are sent in the clear and, even during the rest of the video frame (no matter what's going on with the picture), the horizontal sync pulse is never inverted. It might be weak or missing entirely, but it's never upside down. That's important to keep in mind because if we generate our own horizontal sync, we don't want an upside down, positive-going sync signal present. If that was the case, the two sync signals would add together and cancel, which is not a good thing.

We've talked about how to regenerate sync where the signal being received is unreliable. Basically, the approach is to take the horizontal pulses sent in the clear during the vertical interval and use them as the reference for a phase-locked loop that will supply the missing pulses during the rest of the video frame. If you've got twenty or so reliable pulses per frame, you can accurately generate the missing two hundred and forty or so for the rest of the frame.

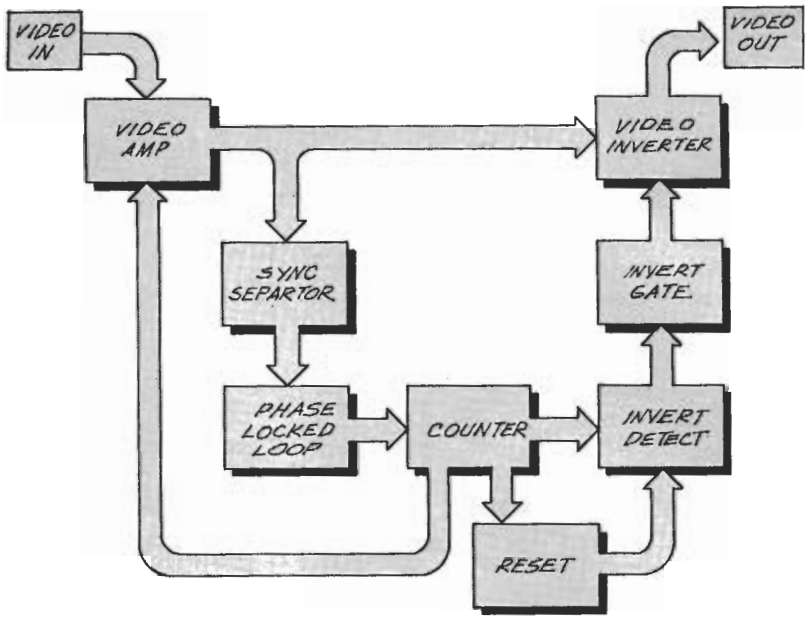


FIG. 9—BLOCK DIAGRAM OF OUR SSAVI DESCRAMBLER. The individual sections of the circuit are not complex.

The block diagram of the circuit we need is shown in Fig. 9. In a nutshell, the job of the circuit is to make sure the picture is always present at the output in a non-inverted state, and that it has horizontal sync pulses present at the right level and the right position.

The scrambled video is fed to an op-amp and the output is sent to a sync separator—the same basic circuit that's found in every TV set in the universe. The sync pulses drive a phase-locked loop whose output is decoded to provide the missing sync pulses for the video lines outside the vertical interval (where most of the interesting stuff is found). These generated sync pulses are mixed with the incoming video and then sent, through a gated inverter, to the back of your TV set.

The gated inverter is controlled by a signal that tells it whether or not the picture portion of the video is upside down. The control signal is derived by watching the state of line 20, as we discussed before.

All this sounds incredibly complicated but, if you look over the block diagram, you'll see that it's just a collection of gates and counters—the same sort of stuff we've been messing around with for years.

Our Descrambler Takes Shape

Building a circuit that can make sense out of SSAVI-encoded signals isn't simple, but it's not impossible, either. Best of all, it can teach you a tremendous amount about basic video, too. We've reached the point where we start turning to hardware. If you look over the block diagram in Fig. 9, you'll see that the circuit we'll need is not complex.

The final thing we talked about was a reset pulse that's needed to initialize the various line counters that will be part of the SSAVI descrambler. We need to find something in the scrambled signal that's stable enough to use as a reset for our digital circuitry.

Remember that everything in the vertical interval is sent "in the clear." One of the components there is vertical sync—an ideal candidate for generating a reset pulse. When you look at scrambled video on a scope or waveform monitor, you may wonder how anything can be picked off the signal. (Incidentally, you stand a much better chance of successfully viewing the scrambled signal if you have a dual-channel scope. Feed standard video into one channel, use that for the trigger, and view the scrambled stuff on the scope's other channel.

Scrambled video may look like a mess, but even broadcast video that's sent in the clear is incredibly jittery. It's a tribute to TV designers and the video standard in general, that the TV set can lock onto anything that comes in over the airwaves.

If you tune your TV to a scrambled signal, you'll note that although the picture is messed up, the screen always shows a full frame. That's because, even though horizontal sync has been altered by the cable company, vertical sync can still be recognized by the circuitry in your TV.

The first piece of hardware we built was a simple demonstration circuit that enabled you to mess up the horizontal sync signal. The first thing we have to do to the video signal to descramble it is separate the sync from the picture.

The circuit shown in Fig. 10 will take video in at one end and give you two versions of the composite sync part of the signal out the other end: positive- and negative-going. The transistor is working as a simple buffer and, by adjusting the video level at its output, we can have the incoming negative sync fall below the high threshold of the TTL EXCLUSIVE-OR (XOR) gate. The first gate produces the composite sync and the second gate works as a simple inverter.

There are other ways to separate sync, but this one has the advantage of giving you an output that swings close to the supply rails, has a very low noise component, and is at TTL logic levels, which makes it much more reliable for feeding the digital circuits we'll be designing for the rest of the descrambler.

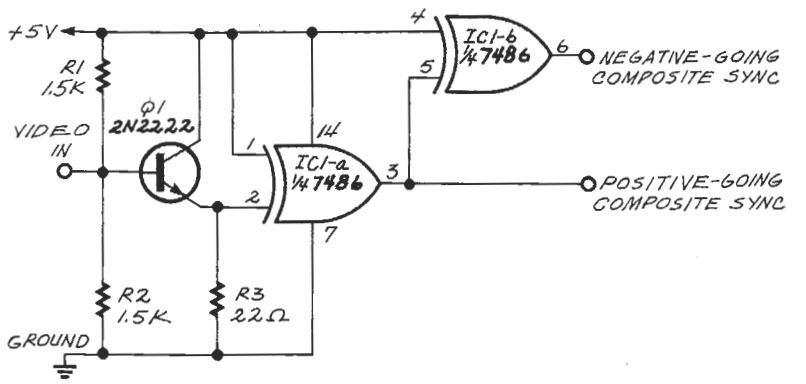


FIG. 10—THIS CIRCUIT WILL TAKE VIDEO in at one end and give you positive- and negative-going composite sync signals at the other end.

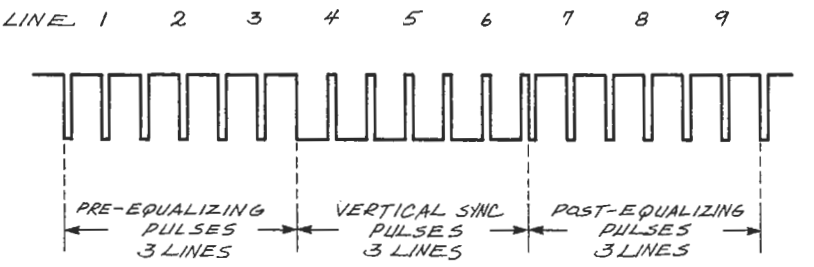


FIG. 11—COMPOSITE SYNC WAVEFORM. Vertical sync is the most negative part of composite sync.

While we're looking at the composite sync signal, this is a good time to work out the details of the reset circuit since it has to isolate vertical sync from the composite sync signal. The way to do that should be obvious when you look at Fig. 11, the composite sync waveform. Just as it's supposed to be, vertical sync is the most negative part of composite sync. To isolate the vertical sync, we need a simple low-pass filter; a suitable one is shown in Fig. 12.

The two gates after the filter clean up the sloppy waveform produced by the R-C circuit. You'll notice that CMOS 4049 inverters square up the shoulders of the waveform. The low-pass filter (or vertical integrator, as it's sometimes called) is being fed with a positive-going version of composite sync and, since it's going through two inverters, it's producing a positive-going vertical sync pulse at the output of the circuit.

That's necessary because we a positive-going vertical sync for the rest of the circuit. As with most things electronic, there are several ways to do the same job, but bear with me until we've gone through the whole design before changing things around. Once you understand the circuit in its entirety, you can start modifying it to your heart's content.

Even though we haven't completed the design of the descrambler yet, the pieces we've finished can be put together as shown in Fig. 13 to produce some interesting and extremely informative waveforms. Video goes in at

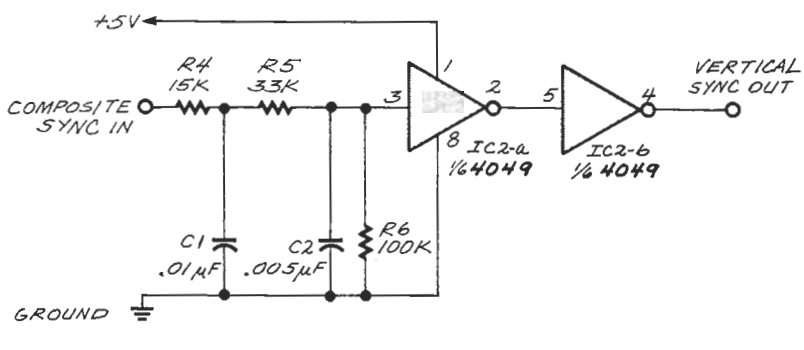


FIG. 12—TO ISOLATE VERTICAL SYNC, we need a simple low-pass filter like this.

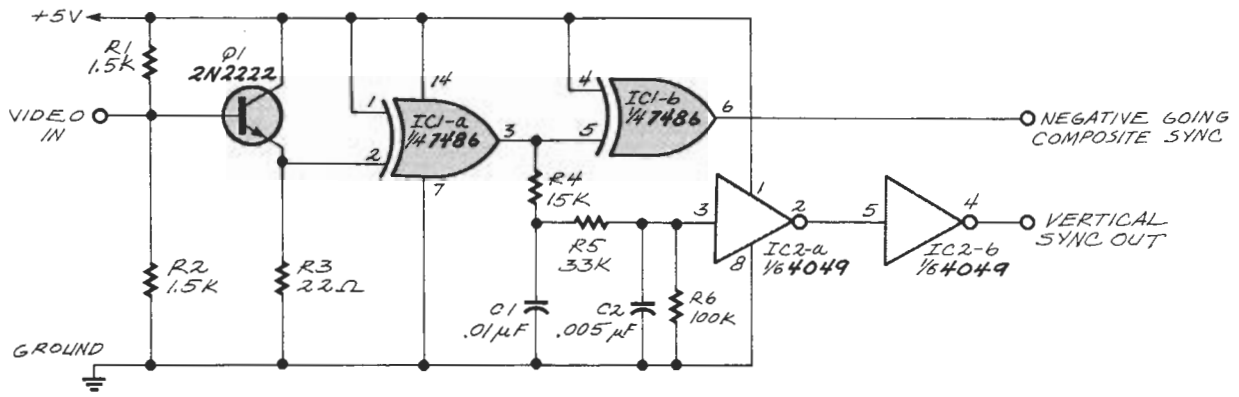


FIG. 13—THE FINISHED PIECES OF OUR DESCRAMBLER can be put together as shown here. Video goes in at one end and we're able to isolate the sync pulses at the other end.

one end and we're able to isolate the sync pulses at the other end. I leave it to you to imagine what a bit of creative gating can do—especially if you use these signals to control the switches in a 4066 as we did in the demonstration circuit in Fig. 7.

Now that we have vertical sync isolated, the next job to do (and the most critical for the descrambler) is to come up with a way of producing horizontal sync. That is obviously more difficult because we know that it won't be present all the time in the received video signal. As a matter of fact, it's a lot better if we operate under the assumption that it's never there at all.

PPL Lends a Hand

Once upon a time, when gated sync was the last word in video signal scrambling, it was relatively easy to descramble the system, as there were two constants you could count on. The first was that the missing horizontal sync pulses were recoverable from information that was buried elsewhere in the RF signal. The second, more subtle, constant was that the horizontal sync pulses were always missing from each line of video.

The SSAVI system makes this part of the descrambling process a bit more difficult.

As we've discussed, the horizontal sync pulse in the SSAVI system is considerably sneakier than in any previous scrambling technique. In any given field of video, the pulses can be absent or at the wrong levels. They could be present, although the chances of that are small. The only constant in the SSAVI system is that the horizontal pulses will be there during the vertical interval—and that's while the electron beam is off the screen.

Given all that, our job is to come up with some way to generate horizontal pulses only when they're needed. Not only that, but we have to be sure that the pulses we create are placed correctly on each line, and are produced at the exact same rate as the horizontal frequency of the incoming scrambled video signal. This sounds like an insurmountable design problem but, in fact, it's not really that difficult. The key to the design is the use of a phase-locked loop, or PLL.

Before we get into the details of how a phase-locked loop circuit is going to solve our sync problem, it's worth spending a few minutes on the basics of phase-locked loops. Since this is such an important part of our total circuit, it's impossible to understand how the descrambler works without a foundation in the theory of phase-locked loops.

The basic components of a standard phase-locked loop are shown in Fig. 14. There are two basic parts: the first is an input conditioning circuit that cleans up the signal applied to the phase detector, and the second is a local oscillator whose frequency is determined by a control voltage. That part is usually referred to as a voltage-controlled oscillator (VCO) or voltage-to-frequency converter.

The output of the VCO is compared to the input frequency, and the phase detector generates an error voltage that's proportional to the difference between the two frequencies. The error voltage controls the frequency of the VCO, and the result is that the VCO's output is always in-phase—or synchronized, if you prefer that term—with the input frequency.

By setting the VCO's base fre-

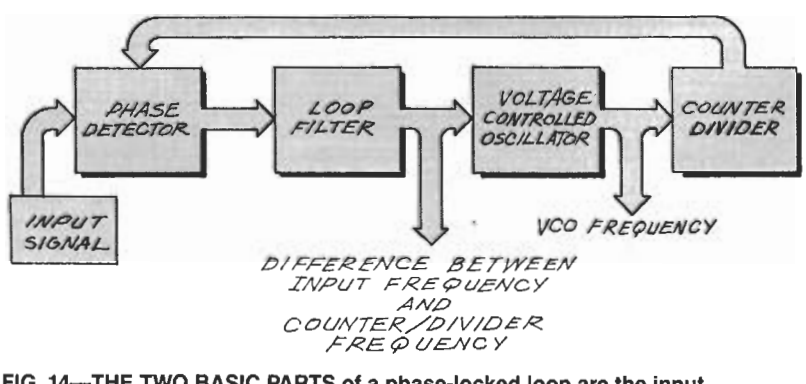


FIG. 14—THE TWO BASIC PARTS of a phase-locked loop are the input conditioning circuit and the voltage-controlled oscillator.

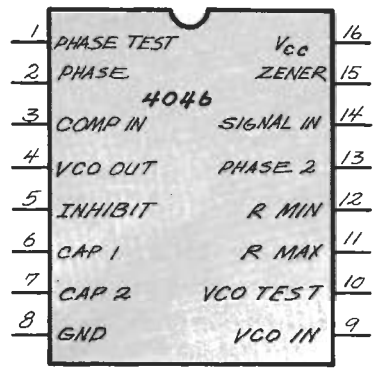


FIG. 15—PINOUT DIAGRAM for the tried-and-true 4046 PLL.

quency to some multiple of the input frequency, we can have the counter/divider chop the VCO frequency down to the input frequency and keep the VCO in sync with the input, even though the frequencies aren't the same. So, PPLs let us easily multiply frequencies, build filters, and—more to our point—keep signals in sync.

Phase-locked loops are basic building blocks in circuit design, and it's well worth your time to learn as much as possible about them. I strongly urge all of you out there to read about, experiment with, and build PLL circuits. There's a lot of good information around regarding phase-locked loops and a good deal of it comes from the semiconductor manufacturers themselves. I know that Signetics has a whole data book devoted to phase-locked loops. Give them a call (408-991-2000) and find out how you can get a copy of their "Phase Locked Loop Handbook." While it's not really the kind of reading that will keep you up at night, it's a very good source of information.

The phase-locked loop we'll be using is the old tried-and-true 4046. The pinout for the chip is shown in Fig. 15, and a block diagram of the chip is shown in Fig. 16. The frequency of the VCO is determined by the RC constant of the resistors at pins 11 and 12, and the capacitor betweens pins 6 and 7. A second factor affecting the VCO is the control, or error voltage on pin 9.

The VCO will stay in sync with the input frequency that's applied to pin 14. If you put a divider circuit between the VCO output on pin 4 and the comparator input on pin 3, the VCO frequency will be the input frequency multiplied by whatever value you're using for the division.

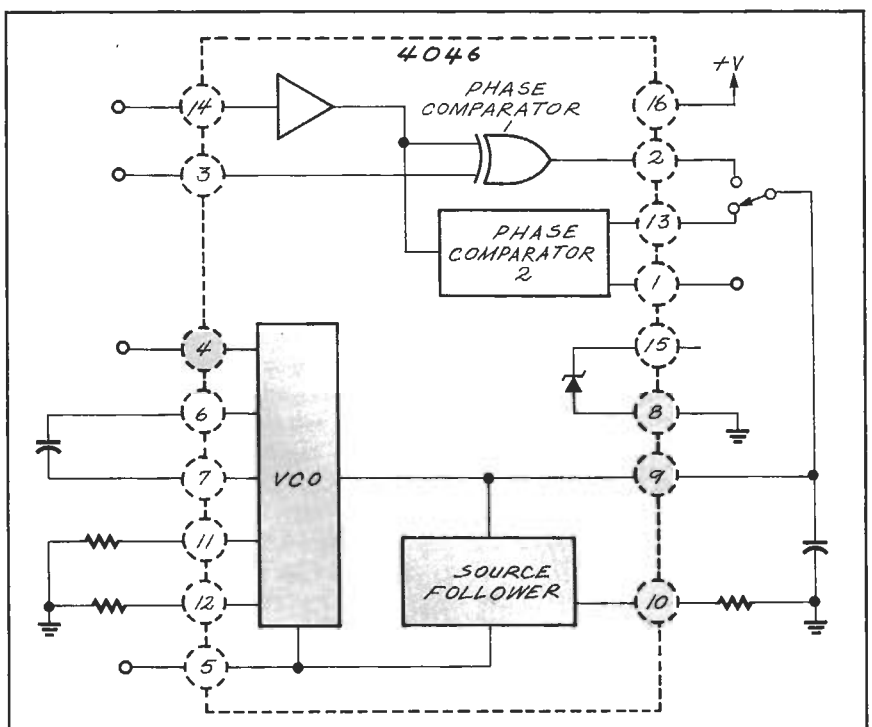


FIG. 16—4046 BLOCK DIAGRAM. The frequency of the VCO is determined by the resistors at pins 11 and 12, the capacitor between pins 6 and 7, and the control voltage on pin 9.

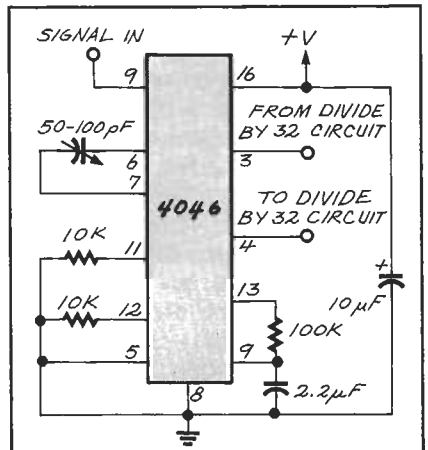


FIG. 17—4046 PLL SETUP. The VCO base frequency is set at 504 kHz, which is the horizontal scan rate (15750 Hz) multiplied by 32.

There are lots of things to watch out for when you're designing a circuit around a 4046, or any PLL for that matter, but we're more interested in the application than the theory.

In our descrambler, the PLL is the perfect solution for solving the horizontal sync problem. Remember that the only time we can be sure of receiving transmitted sync pulses is during the vertical-blanking interval. The question we had to answer is how any circuit could "know" when to generate a horizontal sync pulse if there's nothing that can be used as a reference. The way to make that happen is to do a couple of creative things with a PLL. To start off with, the 4046 setup we need is shown in Fig. 17.

The VCO base frequency is set at 504 kHz. That frequency is an even multiple of the standard horizontal scan rate (15,750 Hz × 32). During the vertical interval, we get 26 usable horizontal sync pulses from the broadcast signal. When line 27 comes along, the picture starts and the horizontal sync is missing. But because the VCO is still running, the divider produces a horizontal sync signal anyway. The pulse is fed back to the input video amplifier and injected into the video signal so that line 27 is displayed correctly on the TV.

The artificially generated sync signal is then split from the video signal by the sync-separator circuit and routed to the PLL. The 4046

has no way of knowing that the sync pulse isn't a "real" one, so it treats it exactly the same as one obtained from the television broadcaster. This kind of self-bootstrap operation continues for the rest of the video frame until the next vertical interval is reached, when the whole thing starts all over again.

As you can see, the success of this whole scheme depends completely on the stability of the VCO in the phase-locked loop. In fact, while that might seem to be a real concern, it's really much less of a problem than you might think. I'm not going to go into the math, but your TV has a tremendous amount of tolerance, and even a ten percent drift in the VCO frequency won't cause much in the way of noticeable shakiness in the TV picture.

Get the phase-locked loop circuit working and closely examine the horizontal sync pulses on your scope. The demo scrambler we built before is the perfect circuit to shut off the transmitted horizontal sync signals periodically. Do that and then watch the results on the oscilloscope.

When we first started work on the SSAVI descrambler, the block diagram probably looked quite complex. But if you look at Fig. 9, you'll see that most of the work has been done. The most difficult job left for us is to come up with the counter that's needed by the phase-locked loop. It's a job we have to do carefully because the success of the descrambler depends on how well we can maintain the stability of the horizontal-sync signal.

The phase-locked loop in our design is driven by a 504-kHz clock based on an R-C network. That frequency was chosen for two basic reasons: The first is that it's exactly 32 times the NTSC line frequency of 15,750 Hz, and that makes it easy for us to do the division necessary to produce horizontal sync pulses. The second, and somewhat more subtle reason, has to do with the period rather than the frequency.

A 504-kHz clock has a period of 1.98 microseconds. That's interesting because, by using five of the counts, we can get a pulse that has a width of 9.92 microseconds, which is close enough to the NTSC's established standard of 10.7 microseconds for a horizontal blanking pulse.

Before we go any further, I have to comment on my apparent disregard for precision. I've been using language and design techniques that are loaded with phrases like "close to," "pretty much the same as," "in the ballpark," and so on. Now, I'm as impressed with standards as anyone I know, but numbers have to work in the real world as well as on paper. You have to keep things like cost in mind when you're designing a circuit.

The NTSC standards for a video signal have a precision of several decimal points, and every broadcaster in the country spends a lot of time and effort making sure his signal adheres to that standard. The same is true of TV manufacturers, as well. The theory for standardization is terrific but, as is usually the case, what happens in reality is different. All you have to do to see what I mean is put a scope on the video signal received by your TV set; you'll find an overall similarity to the video standard, but you'll be amazed at how sloppy the signal really is. The average horizontal frequency might be 15,750 hertz, but there's considerable jitter from line to line in the signal. The same is true for burst, blanking, and all the other components in the line.

Despite these variations in the signal, the picture that shows up on your TV screen is apparently unaffected by them. The difference between studio-quality video (the stuff on the monitors in the broadcaster's control room) and received video (the stuff you see on your TV) is minimal.

Keeping all that in mind, let's design the rest of the descrambler.

The next thing we need is a divide-by-32 counter to complete the phase-locked loop section of the circuit. After my discussion of the relative importance of precision, you should understand why the phase-locked loop's oscillator is only RC-based, and not crystal controlled: It's just cheaper and easier.

The divide-by-32 circuit (the "Counter" in Fig. 9) can be built in several ways; as a matter of fact, we've designed a whole bunch of

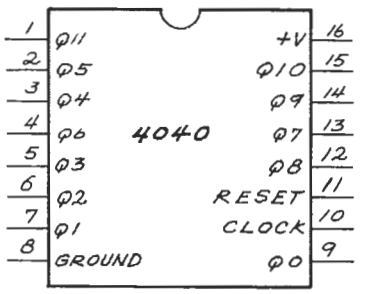


FIG. 18—FOR OUR DIVIDE-BY-32 circuit, we'll use the 4040 binary counter whose pinouts are shown here.

these things in the past. In order to keep the circuit as simple as possible, we'll use the 4040 binary counter whose pinouts are shown in Fig. 18. This is one of the earliest members of the CMOS family and still one of the best choices for general counting. It's a ripple counter, rather than a synchronous counter, so don't use it for applications where super accuracy is required. Remember that a ripple counter is a bunch of sequential counters, and each internal stage uses the output of the preceding stage as an input. This means that the outputs change in sequential order, and an incorrect count will be present briefly on the pins. Since the problem is caused by the propagation delay of each counter stage, the duration of the incorrect count on the outputs is, by and large, a function of the clock speed.

The 4040's clock input is fed with the 504-kHz signal generated by the 4046, and we're using a series of gates to decode the count and provide horizontal blanking and some other timing signals needed for the descrambler. The actual circuit is shown in Fig. 19. You should understand why each of these signals is needed before you start hooking everything up to the back of your TV set. To see what we need from the counter, let's use it to restore horizontal sync and then figure out what else we need to completely unscramble the incoming video.

Getting a horizontal sync clock for the phase-locked loop circuit is simple. All we have to do is pick off the Q_4 output. That is a divided-by-32 version of the input clock from

the phase-locked loop's 504-kHz oscillator and, as shown, it's fed back to pin 3 of the 4046. Now that the phase-locked loop is provided with a constant reference signal, it will accurately generate sync pulses—even when they're not sent along with the transmitted video.

The remaining problem to deal with is turning off the generated sync pulses during the vertical interval when real transmitted sync is present. (Remember that the video isn't scrambled during the vertical interval.) To do that, we have to count the lines of video as they're received and make sure that transmitted sync is processed for the first 26 lines of each frame. The starting point for the count is the vertical sync signal and, as you recall, we've already isolated the signal. The sync separator we built earlier produces a positive-going version of vertical sync. If we use the rising edge of that signal as the zero point for the counter, we have to count a number of lines to reach the point where the lines of video are carrying picture information and are therefore scrambled.

The two lines in the frame that mark the beginning and end of the transmitted horizontal sync are 260 and 27, respectively. We need a circuit that can count the received lines and let the rest of the descrambler know when to use received horizontal sync and when to use the artificial sync generated by the phase-locked loop. The zero point for the counter will be the rising edge of our vertical sync signal. Since that occurs at line 3, we have to decode a count of 24 and 257.

We can use a 4040 to do that job as well. When the counter reaches 24 we have to enable a gate that will send the phony horizontal sync pulses to the video amplifier at the circuit's input. When we get to a count of 257, the gate has to be disabled to let the real horizontal pulses through to re-sync the phase-locked loop circuit.

Decoding those two numbers is a pain in the neck since it involves watching nine counter lines. You can do that with a bunch of gates, but a better way is to use an EPROM. The choice is yours although, if you haven't had much experience in this

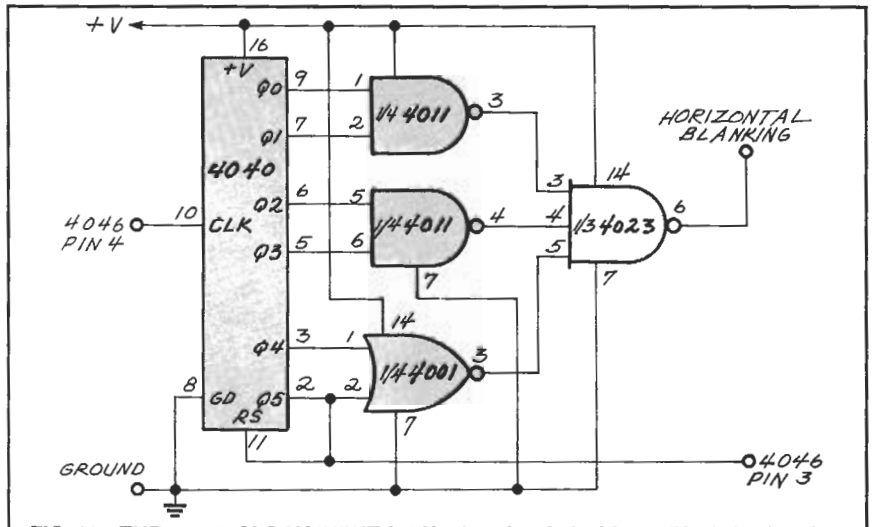


FIG. 19—THE 4040's CLOCK INPUT (CLK, pin 10) is fed with the 504-kHz signal generated by the 4046. We're using a series of gates to decode the count and provide horizontal blanking.

sort of decoding, it's probably a good idea to work out the logic and build the decoder with gates, Otherwise, an EPROM is the way to go. Assuming, of course, that you have programming capabilities.

Correct Sync

You can use any counter you want, and the 4040 is as good as any one of them. The tricky part of the design is that the decimal 257 is 100000001 in binary code. That means we need a counter/decoder that can handle nine lines. The 4040 can output the correct count, but the decoder must be able to "watch" nine lines. That's not a problem if you're using discrete logic gates to do the decoding because you can have as many input legs as you want. If you're using an EPROM, it's obvious that the extra available address lines have to be tied to either power or ground.

Somewhere out there in designland, there's a combination of gates that will decode the two numbers (24 and 257), but finding it is pretty tedious and, to make matters worse, there's not a lot to be learned from doing it. I haven't given it a lot of thought, but it can probably be done with a handful of gates. I'll leave the rest of this as an exercise for some of you out there.

Don't get me wrong—decoding like this is sometimes necessary,

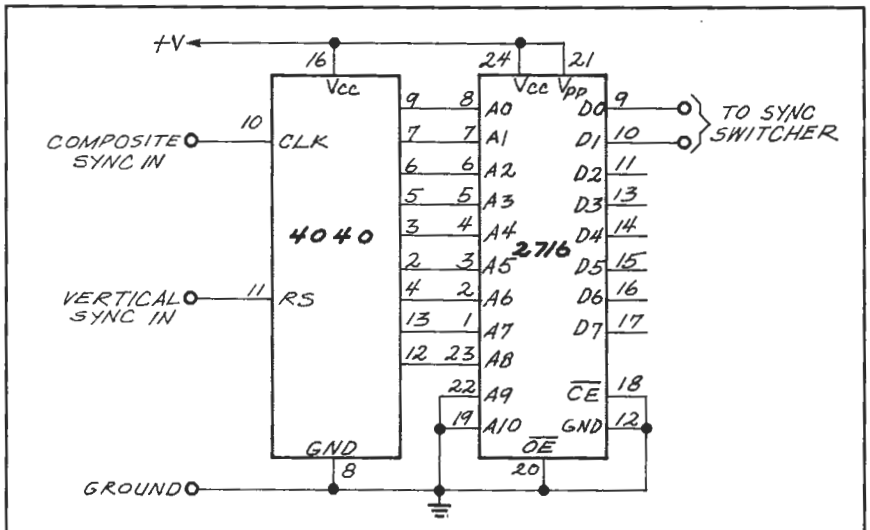


FIG. 20—THE EPROM WILL DECODE two input addresses (24 and 25) with only one or two data lines.

and if you're new to design, it's a good learning experience. But a smarter way to go about this is to use an EPROM, a one-chip solution to the problem.

You can use any EPROM you happen to have around, because we're interested in only two input addresses (24 and 257), and need only one or two data lines (depending on how we design the circuit that enables the manufactured or transmitted sync). The circuit is shown in Fig. 20, and the EPROM's truth table is shown in Table 1. I'm using two data lines to switch between sync sources, but a design could easily be worked out that uses only a single data line.

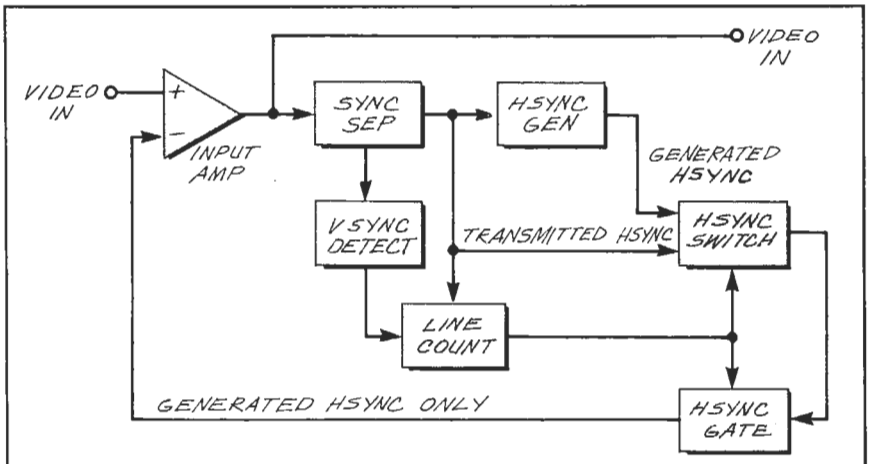


FIG. 21—SSAVI DESCRAMBLER BLOCK DIAGRAM. We use the transmitted sync during the vertical interval and the generated sync for the rest of the time.

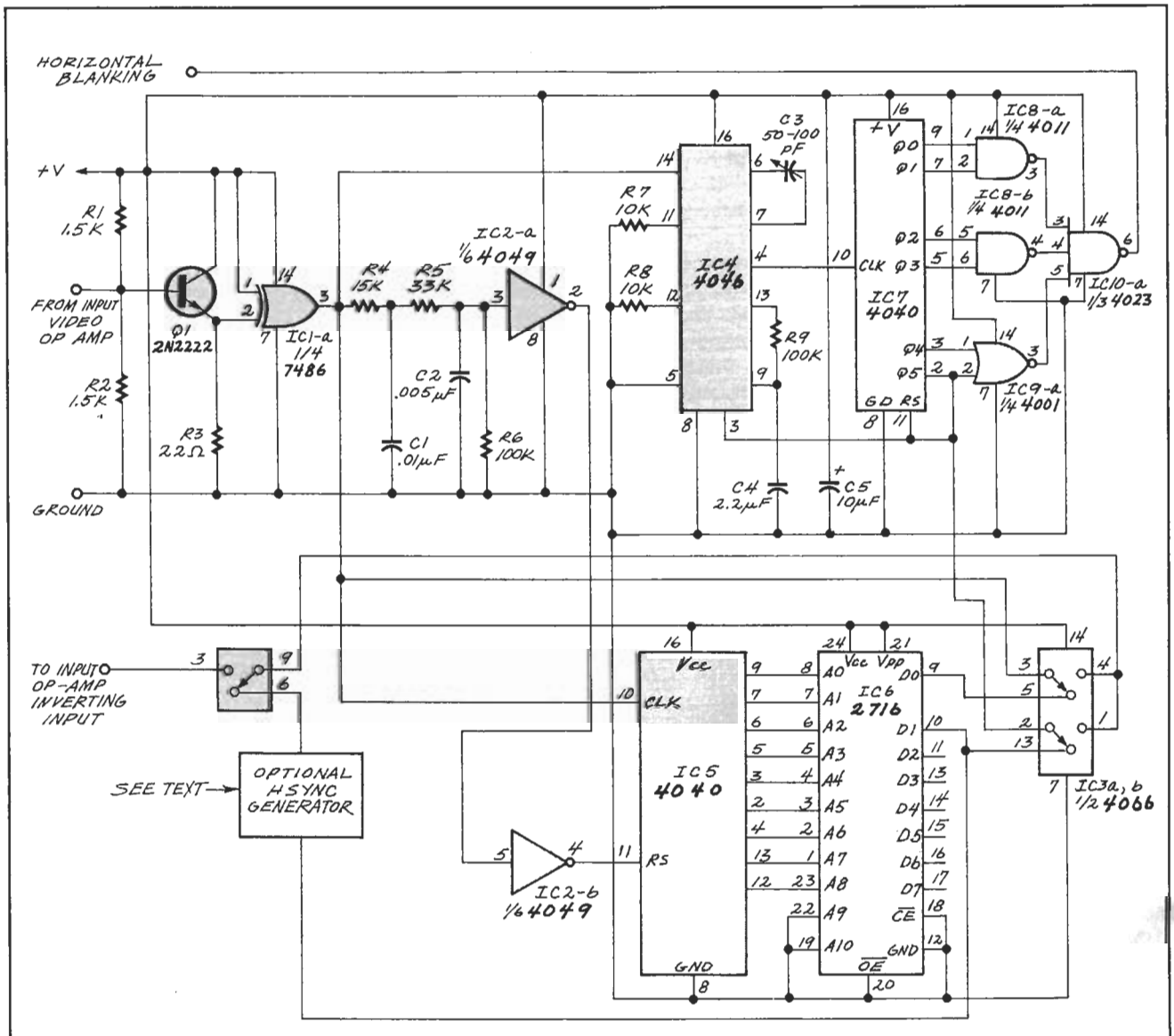


FIG. 22—THE SYNC SWITCHER is just an electronic single-pole, double-throw switch. Here we've added the EPROM and the 4066 switch. The EPROM's data lines directly control the two 4066 control lines.

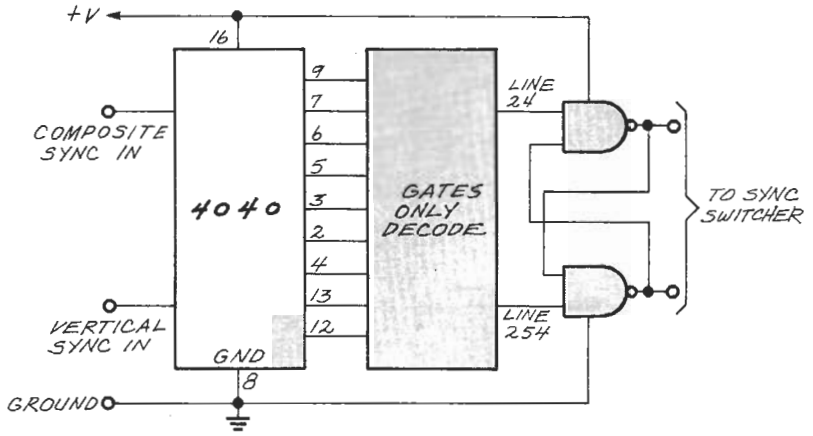


FIG. 23—INSTEAD OF AN EPROM, you can use some of the left-over gates to build a set/reset flip-flop.

This is easier to understand when you look at Fig. 21, a block diagram that shows all of the circuitry we've just been talking about. There are two possible sources of horizontal sync pulses: the ones from the original video signal, and the ones being generated by the phase-locked loop circuit. We want to use the transmitted sync during the vertical interval (it's sent in the clear during the vertical interval), and the generated sync for the rest of the time. We have 28 lines of signal with transmitted horizontal sync; they are lines 261 and 262 at the end of one frame, and lines 1–26 at the start of the next frame.

The sync switcher we need is really just an electronic version of a single-pole, double-throw switch, and the easiest way to put one of them together is to use a 4066. Figure 22 shows our circuit so far, with the addition of the EPROM and the 4066 switch. Notice that the EPROM's data lines directly handle the two 4066 control lines. That can be done because the EPROM outputs change state only when the 4040 counter signals the arrival of either line 24 or line 257. If you use gates to decode the counter output, you'll have to find a way to do the same thing. One approach would be to use some left-over gates to build a set/reset flip-flop whose control lines are triggered by the arrival of lines 24 and 257. The basic idea to effect this desired action is outlined in Fig. 23.

TABLE 1—EPROM CHARACTER GENERATOR CHART

	Input	Programmed Output Data								
Line Number	EPROM Address	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	Hex Data Byte
1	000	0	0	0	0	0	0	0	1	1
2	001	0	0	0	0	0	0	0	1	1
3	002	0	0	0	0	0	0	0	1	1
4	003	0	0	0	0	0	0	0	1	1
5	004	0	0	0	0	0	0	0	1	1
6	005	0	0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
24	017	0	0	0	0	0	0	1	0	2
25	018	0	0	0	0	0	0	1	0	2
26	019	0	0	0	0	0	0	1	0	2
27	01A	0	0	0	0	0	0	1	0	2
28	01B	0	0	0	0	0	0	1	0	2
29	01C	0	0	0	0	0	0	1	0	2
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
257	080	0	0	0	0	0	0	0	1	1
258	081	0	0	0	0	0	0	0	1	1
259	082	0	0	0	0	0	0	0	1	1
260	083	0	0	0	0	0	0	0	1	1
261	084	0	0	0	0	0	0	0	1	1
262	085	0	0	0	0	0	0	0	1	1

This EPROM can also be used to decode line 20 (see text) by programming one of the data lines high when line 20 (address OFh in the EPROM) is reached.

Back to Theory

Let's go over the general SSAVI theory for a minute. In the SSAVI system, there are two parts of the video signal that get messed up: the first is the horizontal sync pulse and the second is the polarity of the picture portion of each individual line of video. All the circuitry we've been developing so far has been aimed at taking care of horizontal sync. The circuitry has become a bit complicated, but we now have a way to generate sync even if it has been left out.

The circuitry we've built so far will do a good job of restoring horizontal sync. Just about the only problem you might have relates to the width of the generated pulse. The official width of horizontal sync pulse, according to NTSC specifications, is 4.7 microseconds—and the closer you get to that, the better your chances are of having everything work properly. That leads us to the age-old question, "How close is close enough?"

The answer to that question depends on your TV's horizontal sync detector; some of them will recognize pulses that are as much as 50 percent off, while others will turn up their noses at any deviation beyond 25 percent. If you find that the 4046 pulses aren't triggering the horizontal flyback in your TV, you can use those generated pulses to trigger additional circuitry that will produce a pulse of exactly 4.7 microseconds. Then feed that to the input video op-amp instead of using the 4046 pulses directly to do the job. We designed a circuit to do exactly that when we built a video-sync generator. The circuit is shown in Fig. 24.

Now let's address the problem of inverted video. When we first started this project, we talked about how

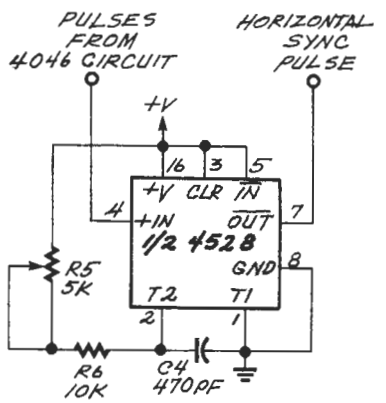


FIG. 24—IF THE 4046 PULSES WON'T TRIGGER the horizontal flyback in your TV, this circuit will generate 4.7-microsecond pulses when triggered by the 4046.

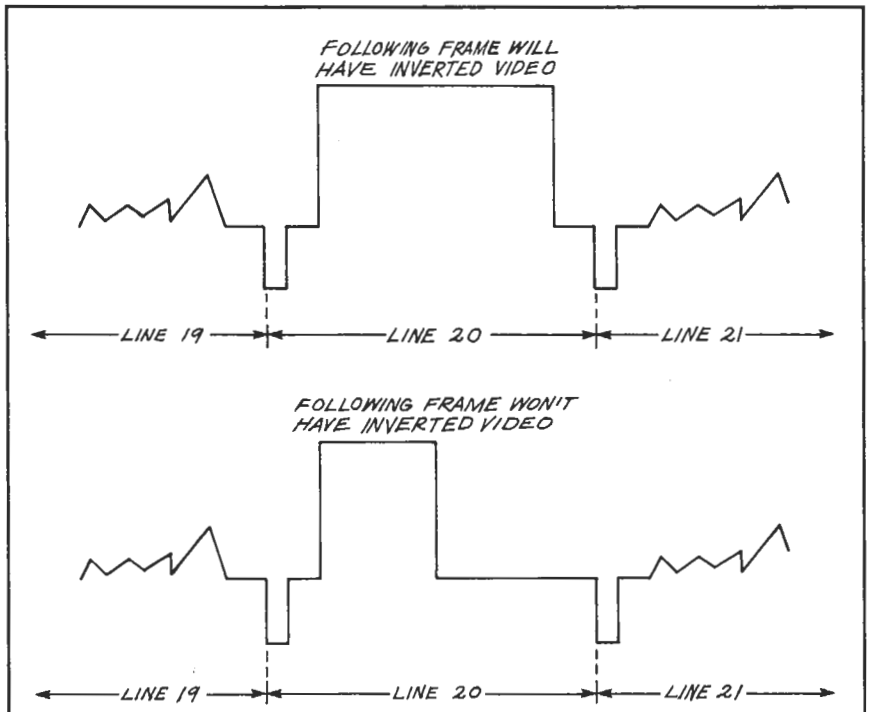


FIG. 25—TO CORRECT inverted video, we need the information buried in line 20.

the SSAVI system encodes information about the polarity of the next frame of video. The original SSAVI system buried this information on line 20, as shown in Fig. 25. Now that we have circuitry to count the lines of video, it's a piece of cake to signal the arrival of line 20 and examine it.

Working with Video

When we built the divide-by-32 circuit (Fig. 26), we added some gates to the output of the counter to create a pulse that was approximately equal to the transmitted horizontal blanking pulse in both polarity and width. I also mentioned that the 2-microsecond period of the phase-locked loop's clock would turn out to be a very useful tool—and you will soon see why this is and what we can use it for.

Since we can identify the beginning of the horizontal blanking pulse, we can use the 504-kHz clock pulses to sample the video at any 2-microsecond multiple along the line. All we have to do is use the arrival of the horizontal blanking pulse (its falling edge) as a starting point, count the desired number of 504-kHz pulses, and sample the video to get the DC levels we want. Picking the points to sample for black-and-white DC levels is critical because they're needed to keep the picture brightness from changing when the signal switches between normal and inverted video.

The best line to use for this sampling is the same one that tells us whether the video is going to be inverted or not—I'm talking about line 20. As you can see in Fig. 27, when this line appears, the black level can be read from the back porch, and the correct white level can be gotten from the first half or so of the picture portion of the line (immediately after the rising edge of the horizontal blanking pulse).

Because the clock pulses from the phase-locked loop are slightly less than 2 microseconds wide, a single pulse sample can be taken four clock pulses after the leading (falling) edge of the horizontal blanking pulse to lock in the black level. The white-level sample can be taken two clock pulses after the end of the horizontal blanking interval (the rising edge of the pulse); it can be as

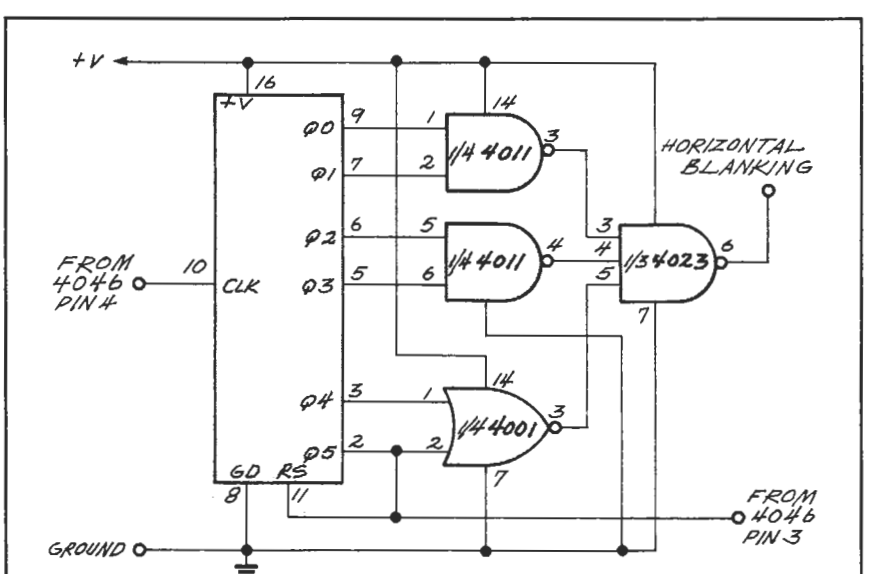


FIG. 26—THE GATES WE ADDED in the divide-by-32 circuit create a pulse that's approximately equal to the transmitted horizontal blanking pulse in both polarity and width.

much as four clock pulses wide. You can understand this by examining Fig. 28.

After using line 20 to establish the correct DC levels for the following frame, we have to sample a portion of the last part of the line to see whether the picture will be inverted or not. A reasonable location to pick is about 50 microseconds into the line, which would be some 25 clock pulses after the leading edge of the horizontal blanking pulse. If the sample taken is high, we know that the next frame is going to be inverted—if the sample is low, the frame will be normal.

All this may sound complex but, if you think for a moment, you'll realize that most of the needed circuitry has already been designed. The low-order output (Q_0) of the 4040 that's counting the 504-kHz pulses from the phase-locked loop is giving you a series of 2-microsecond pulses, and the gates hanging off the 4040 outputs are producing a synchronous analog of the horizontal blanking pulse. To sample the line as I just described, all you have to do is detect the leading or trailing edge of the blanking pulse (whichever one you need), count up the appropriate number of 2-microsecond pulses, and sample the video line.

All the signals you need to determine the DC levels and polarity of the following frame can be obtained by decoding the outputs of the 4040. That can be done using the same techniques we used earlier to recreate the horizontal blanking pulse. Since the period of a line of video is about 64 microseconds, and we have a clock pulse with a period of 2 microseconds, we can think of each video line as being divided into 32 segments. By counting and decoding properly, we can examine any segment of the line in any 2-microsecond multiple width. The polarity sample, for example, would be a 4-microsecond (2 clock-pulse) segment taken 25 clock pulses after the 4040 has been reset to zero. That will also mark the beginning of the decoded horizontal blanking interval.

Once the polarity sample has been taken, the same technique used to switch between generated and transmitted sync can be used to correct the video polarity if we find that it's been inverted. The two unused switches in the 4066 have to be configured, once again, as a single-pole, double-throw switch. The output of the gate that reflects the state of the polarity sample triggers a set/reset flip-flop that, in turn, toggles the SPDT switch.

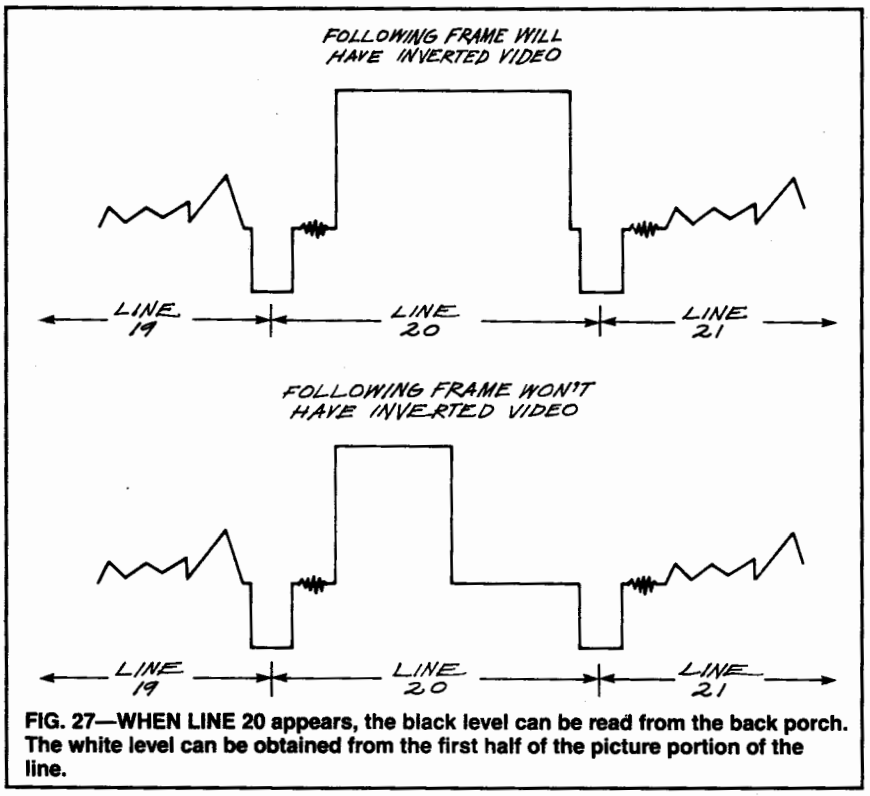


FIG. 27—WHEN LINE 20 appears, the black level can be read from the back porch. The white level can be obtained from the first half of the picture portion of the line.

The output of the amplifier at the front of the circuit contains both the received video and, except during the vertical interval, the generated horizontal sync signals created to stabilize each line of video. If the polarity wasn't a problem, we could send the signal off to the output stage of the descrambler and then directly to the back of a TV set.

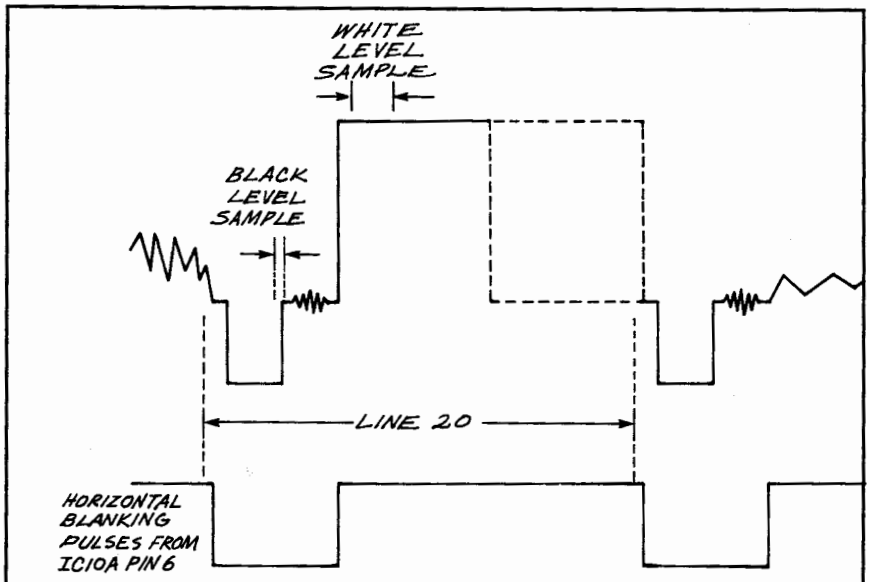


FIG. 28—TO LOCK IN THE BLACK LEVEL, a sample can be taken four clock pulses after the leading edge of the horizontal blanking pulse, and the white level can be taken two clock pulses after the end of the horizontal blanking interval.

Since, however, the "VI" part of SSAVI stands for "video inversion," we have to have available an inverted version of the video as well.

The received video with the corrected horizontal sync pulses is sent through an inverter that's built in exactly the same manner as the single-transistor buffer at the front of the decoder. This inverted version of the video is sent to one side of a polarity switch, and the plain video is sent to the other side of the switch. Each signal has its horizontal interval corrected by the circuitry that was designed to restore the sync signals.

The flip-flop built to indicate the presence of either normal or inverted video controls whether the normal or inverted video is routed to the descrambler's output.

Even though the descrambler's circuitry seems to have grown at an alarming rate, its overall operation is not hard to understand. If you've been following this from the beginning, you should have a pretty good handle on what's going on. We've regenerated horizontal sync and created a signal to tell us if the picture has been inverted. At the descrambler's output, we're using an electronic switch to make sure the video sent to the back of your TV set always has the correct polarity. This has been done by decoding the state of the transmitted video from line 20, and using that information to channel either a straight or an inverted version of the sync-restored video to the output of the descrambler.

The last piece of hardware business comes up because the horizontal sync is *never* inverted in the SSAVI system. Since the first thing we did to unscramble the video was to restore the horizontal sync pulses, the video coming out of our inverter (the video being sent to the invert side of the 4066 switch) will have the entire horizontal interval inverted as well. Fortunately, this is pretty easy to correct. We have to make sure that during the horizontal interval, only the non-inverted video is routed to the descrambler output.

The way to accomplish this is to gate the output of the polarity indicator (the flip-flop) with the horizontal blanking pulse. We want the normal video signal sent to the descrambler output during the horizontal blanking interval, regardless of the polarity of the video signal.

When you're building a SSAVI descrambler like this, there are a few rules to keep in mind as you work your way through the design. We've gone over all of them, but listing them out will make it much clearer:

1. Since the vertical interval is always sent in the clear, the descrambler has to be disabled for this period of time.
2. During the horizontal interval, the transmitted video signal must be sent to the descrambler output.
3. If you don't have a scope, you won't be able to build a SSAVI, or any other video descrambler.

Go Build Your Own

You now have all the information you need to design and build a working descrambler. I admit that it's been a lot of work, but it's a low price to pay because the descrambler, when combined with the service manual for your TV, will let you use all the features in a cable-ready set.

I started writing on this subject because my local cable company began scrambling *all* the cable channels, not just the premium ones. A lot of people, myself included, have spent a lot of money to buy a TV that offers picture in picture, super stereo, and a bunch of other features. Using a box from the cable company completely wipes out most of those features, to say nothing about not being able to use the TV's remote control.

Because cable boxes are fairly expensive, even when purchased in quantity by the cable companies, the boxes provided with cable service don't provide the same range of functions as those built into a high-end TV set. Some boxes have audio and video outputs but, at least in my area, the audio is in mono—I didn't even know that mono was still a viable option.

The mentality of the cable companies today is on a par with that of the phone company twenty years ago. Not only were you charged for line service, but the phone company also insisted on billing you for all the hardware, wire, and installation as well.

It took a lot of legal work and lawsuits to convince the phone company that it was more profitable to concentrate on selling services than it was to keep a stranglehold on everything from your mouthpiece to my earpiece. For once, a bunch of lawyers did something that was worthwhile!

Until the cable companies learn the same lesson that was force fed to the phone company, we're going to have to deal with the problem of the limitations of cable TV's proprietary hardware vs. the cable-ready TV set. If basic cable service includes 25 non-premium, non pay-per-view stations, I shouldn't need a cable box to get those channels. The big-brother, total-control mentality of the cable television companies is going to be ended, I fear, only by lawyers.

Real-world Considerations

In the original SSAVI system, horizontal sync was never inverted during active video (when the picture was showing up on the TV), but that has changed. The video waveform in Fig. 29-*a* shows the state of the horizontal interval as of fifteen minutes ago (as of this writing) on my cable. I've drawn it with an unscrambled horizontal interval so you can see how the scrambled signal relates in time to the normal one in Fig. 29-*b*. A change has been made to the 4-7-microsecond position normally occupied by the horizontal sync signal. There are also two 1-microsecond spikes at the very beginning and end of the horizontal interval. If you watch a scrambled picture, you can see these at both ends of the interval as it weaves its way down the middle of your screen.

These spikes peak at 100 IRE units but since they're not really in the horizontal interval, they don't cause any problem. If I had to make a guess as to why the "woop-dee-doo" has been added to the horizontal interval, I'd say that it's to keep people from doing what we're doing—adding a single sync pulse to restore the signal.

When I was designing the section of the descrambler that put the horizontal sync pulse back in the interval, I had to modify the circuit slightly to make it work. Basically, all

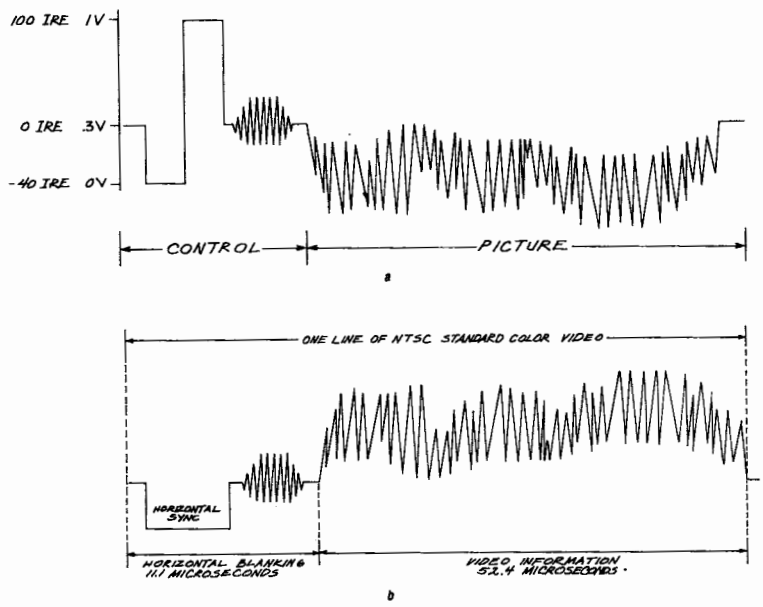


FIG. 29—THIS VIDEO WAVEFORM shows one variation on the SSAVI system. A change has been made to the 4.7-microsecond position normally occupied by the horizontal sync signal, and there are 1-microsecond spikes at the beginning and end of the horizontal interval.

I did was add some gates to the output of the 4040 line counter to create a pulse that started 2 microseconds into the horizontal interval and had a width of 4 microseconds—the approximate length of the horizontal sync pulse. I used it to gate the incoming video, completely eliminate the change from the transmitted signal, and make sure that the only thing that appeared in that section of the line was the generated horizontal sync from the phase-locked loop.

During the vertical interval, of course, this entire activity was disabled to allow the transmitted sync (which is still being sent in the clear) to be passed through the descrambler.

The polarity indicator for the picture still works as I described, but the newer SSAVI systems (at least the one in my area) move it around between lines 20 and 22. This is actually a dangerous thing for cable companies to do, since line 22 is usually considered to be active video.

The information as to where it will be is probably buried in the subscriber codes, which are difficult to decode. The code format is usually as a series of 32-bit words with bits that are about 2-microseconds wide at a data rate of 504 kHz (which should be a somewhat familiar number).

How to handle this problem depends on the nature of the scrambled signal in your area. You can decode the marketing code and figure out which bits indicate the correct line to examine for determining the polarity of the following video frame, but that is an involved subject and there's just not enough room here to go into it. It's also not the best approach since there's nothing stopping the cable companies from putting the code somewhere else or, to make matters even worse, change the encoding algorithm.

A second way to deal with the problem is to examine the vertical interval on a scope, see where the polarity-indicating lines are, and work out some circuitry that examines them all. Remember that what has changed is the location of the line (it now moves around from place to place), and not the structure of the line. In essence, if there are three lines to examine, a high in the second half of any of them would indicate that the next video frame is inverted. That's the approach I took.

One interesting piece of information I can pass along to you is that, again in my area only, the video is *always* inverted. Check the signal in your area and see if that's true for you as well. If it is, the design of the descrambler is much simpler.

Before you get to work, however, some thought has to be given to the things that might be done in the future. Since the SSAVI system has changed over the years, there's no guarantee that the way you go about detecting inverted video today is going to hold up tomorrow.

If you go through the trouble of building something to detect the state of video by looking at line 20, it can all be made useless if the cable company moves the information somewhere else. Restoring horizontal sync is pretty much locked in stone, but I can definitely think of a few ways that even that could be changed.

Remember that large-capacity EPROMs are cheap, and there's nothing stopping the cable company from putting several encoding techniques in the chip. To guarantee your work against obsolescence, the techniques you use to clear up inverted video have to depend on things that can't be easily changed by the cable company.

There's a much more interesting way to deal with the problem.

The vertical interval provides a lot of information. One thing we haven't talked much about is the white and black levels. I don't bother too much with these because the TV (or VCR) has excellent circuits to clamp the levels and condition the video signal before it reaches the sync separator. I've found that if I feed the video in at anywhere from 1.2 to 1.5 volts DC, the TV or VCR doesn't have any truble working out the levels for itself.

The black level (0 IRE) should be about 0.3 volt and the white level (100 IRE) should be about 1 volt. Super black is the bottom of sync (−40 IRE) and is at 0 volts. Even if you can't find the polarity line to get the white level, you can always get the black and super-black levels from the unscrambled transmitted sync that's sent in the vertical interval. You can even get the black level of the video signal in the horizontal intervals during active video from

the 5-microsecond section containing the back porch.

Once you know the black level, you can integrate the picture part of the line to get the average DC voltage of the signal. That's the same sort of thing we had to do to isolate vertical sync from the composite sync signal. If it's below the black level, you know that the line has been inverted, and if it's higher, then you know that the line was normal. You'll probably have to filter out the chroma and look at the luminance (DC level) of the line, or else the circuit might have a hard time telling the difference between a dark normal video line and a bright inverted line.

If you do that, one other clue that will help detect an inverted line is that the picture's DC level should never get below about −20 IRE. If it does, the TV's vertical sync detector might sense it and falsely trigger a vertical retrace. That's the sort of thing that frequently happens when you try to view a scrambled video signal—even if the transmitted signal isn't doing anything to the original vertical interval.

Since video is inverted (or not) by the cable companies on a frame-by-frame basis, it's safe to assume that finding an inverted line means that the next one will be inverted as well. Remember that the descrambler is completely reset during the vertical interval when vertical sync is detected.

There's no way that I can provide you with an absolute method to descramble the video in your area—there are just too many subtle variations that can be added to the basic SSAVI system. The ones I just described are only a few of the many possible twists that can be done to a video signal.

The approach and circuitry we developed over the last few months—as well as the theory—should give you a good head start for working out the details that have to be added to deal with the particular scrambling method used by your cable company.

Some Reader Suggestions

Over the years, a lot of the projects we've worked on together have needed oddball decoders. I've said over and over again that my preferred solution is an EPROM. I've used EPROMs for everything from custom character generators to state detectors for weird numbers. If you've got the time and patience to work out a gates-only solution, you might improve your logical thinking skills, but it will take you a lot longer to get something working, it will make PC board layout a lot more complicated, and it will lock you into a particular design. EPROMs are more versatile because any modifications to the hardware in the design can be accommodated simply by programming some new code in the EPROM.

When you're in the middle of designing some hardware, a gates-only decoder might seem more attractive if you can't program an EPROM right then and there. But if you do a lot of hardware design, an EPROM programmer is a piece of equipment that's just as essential as an oscilloscope.

I'm mentioning this because since we went through the basics of a SSAVI descrambler, I've received a lot of mail with alternatives to the EPROM decoding scheme I used to detect lines 24 and 257. Since it seems that a lot of you out there either prefer to do stuff with gates or don't have access to EPROM programmers, I'm going to pass along some of the decoders I've received.

All the decoders that were sent in are built with standard gates, so you should have no trouble getting the parts. Even though I have the greatest faith in my readers, I'd be a bit remiss if I didn't tell you that I haven't tried these circuits myself. You should experiment with them before you lock them into your decoder design.

The first one is from David Siegel of Livonia, Michigan and the schematic is shown in Fig. 30. It's a pretty slick design in that it's built with only three chips: two dual 4-input NOR gates and one dual 4-inch AND gate.

The second decoder is from Chris Carson of Ottowa, Ontario. His design is a bit more complicated, but that's ok. Remember that more complexity makes a design more interesting. As you can see in

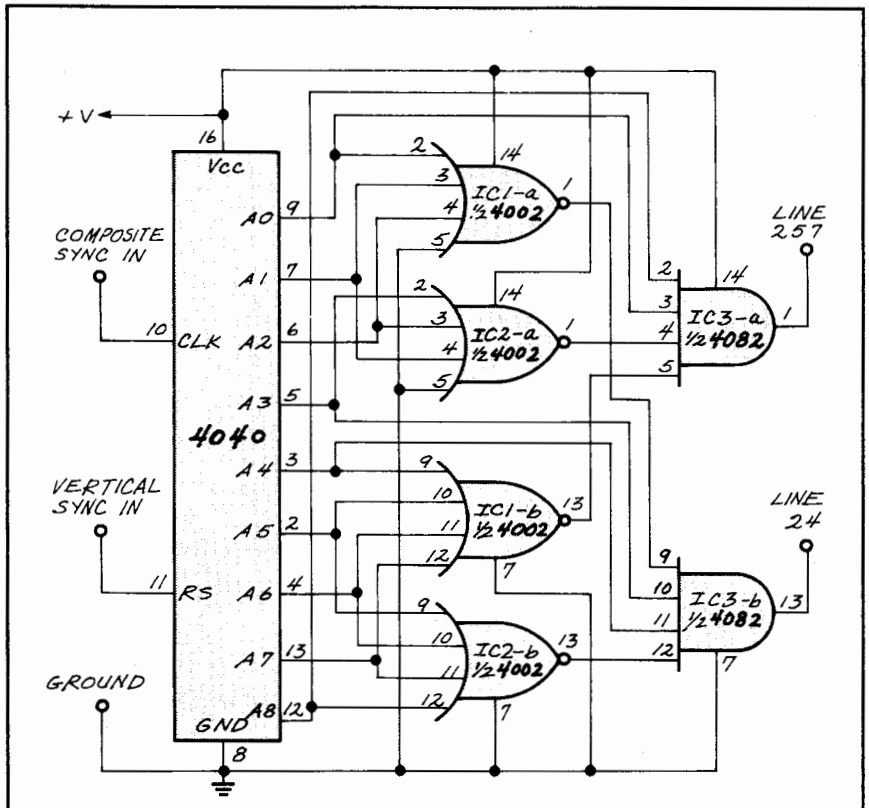


FIG. 30—MANY READERS SENT IN ALTERNATIVES to the EPROM decoding scheme I used to detect lines 24 and 257. This one uses only three chips.

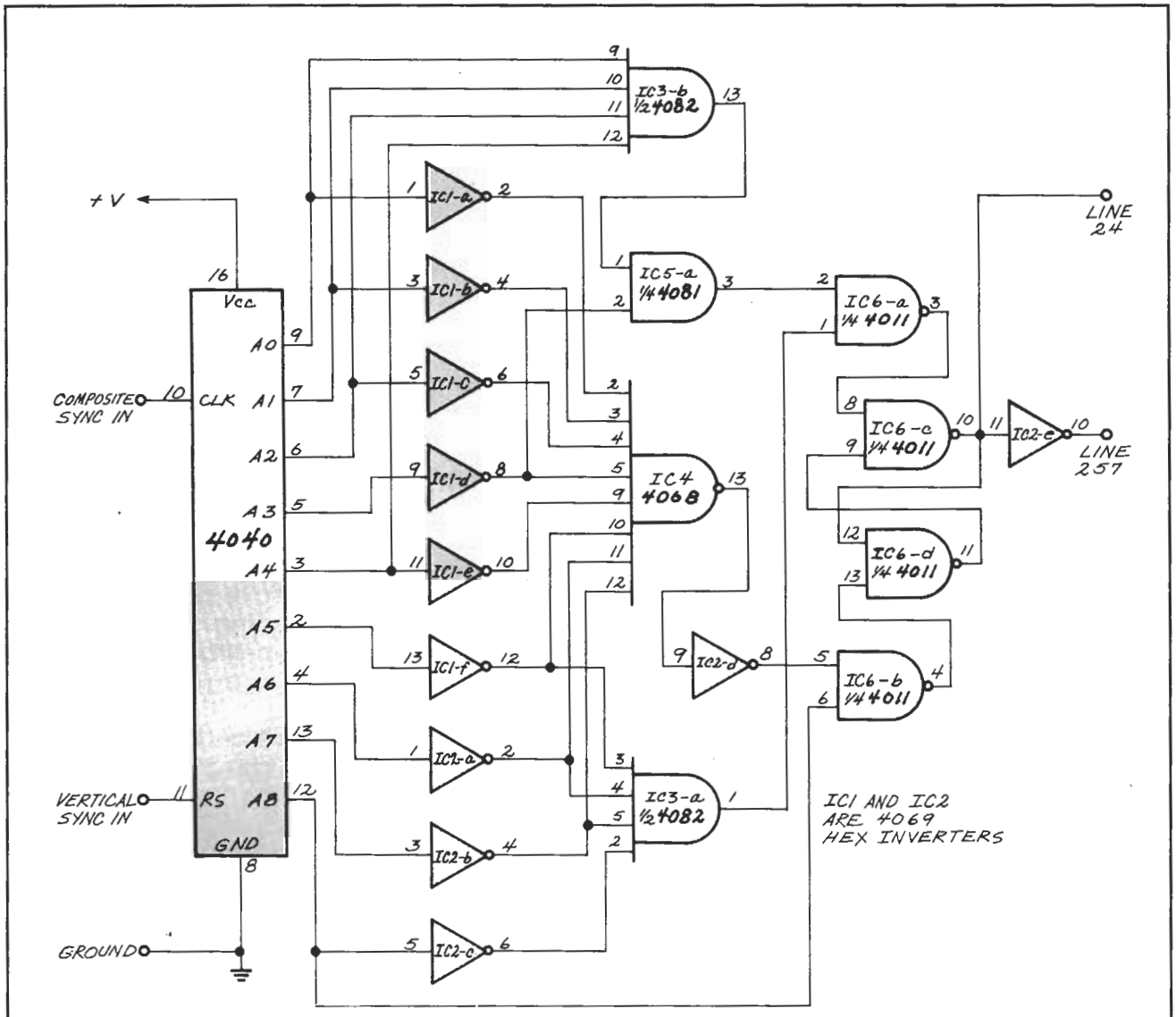


FIG. 31—THIS DESIGN IS MORE COMPLICATED, but only one pin is used for the line indicator, which is handy if the rest of your descrambler wants the start and end of the vertical interval to be indicated on a single line.

Fig. 31, one nice feature is that only one pin is used for the line indicator. That can be handy if the rest of your descrambler wants the start and end of the vertical interval to be indicated on a single line. ■