A PC-Based Digital Storage Oscilloscope and Logic Analyzer

Build this handy multipurpose instrument for your test bench.

By Larry Cicchinelli, K3PTO

I believe that most Amateur Radio operators today will agree that our hobby is somewhere in the middle of the "digital revolution." As such, we are using digital techniques more and more in our daily communications. The rapid growth and acceptance of PSK31 may be considered an indicator of this revolution. Those of us who like to "roll our own" and experiment with our equipment recognize the importance of having good test equipment.

The device described in this article is a combination Digital Storage Oscilloscope (DSO) and Logic Analyzer (LA). It interfaces to a PC via a parallel (printer) port and is built using several printed-circuit boards. The system is designed such that it allows for six input cards. Each card can be either eight logic-analyzer channels or a single analog channel (8-bit resolution).

I am considering designing a serial interface for the device—if there is enough interest from readers. However, I have not yet decided on the exact mechanism yet. It will probably be based on a microprocessor. The current parallel-port interface can access the 130 kbytes of data in about 1 second per board. Unless the PC and microprocessor serial ports can achieve 920k baud it will be somewhat slower. A serial interface would make this device available for use on those operating systems that do not allow direct access to the parallel port.

A previous article (*QEX*, Jan 2000) describes an eight-channel LA that I built and have used quite a bit. The knowledge and experience gained in its development were instrumental in the development of this project. The original intent was to build a DSO only; however, it soon became obvious that most of the design was also useful for implementing a logic analyzer (see Fig 1). I used many of the ideas from the original LA in the design of this combination device. My main goals for the DSO/LA were:

- Ease of construction—use printed circuit boards
- Two analog channels
- Minimum 20-MHz sample rate
- LA channels
- Selectable pre-trigger count
- Trigger selection to allow any combination of analog, digital and LA signals

119 River Run Cir
Sacramento, CA 95833
k3pto@arrl.net

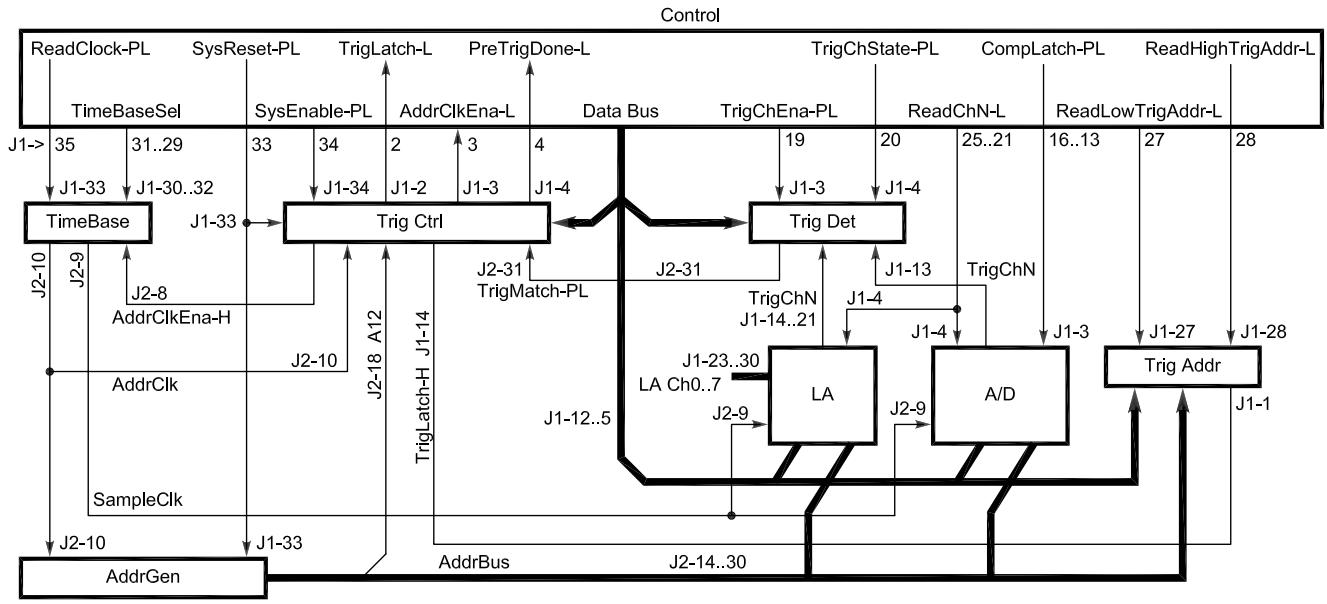

Fig 1—A block diagram of the DSO/LA project as separate boards. (This diagram should be used for reference only—the PLD implementation combines several of these boards onto one.)

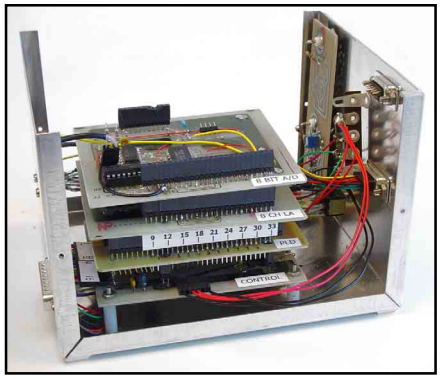

Fig 2—A picture of my "sandwich" construction before I converted several circuits to PLDs.

- An initialization file (referred to in the text as the INI file) that allows the user to define almost all of the operating parameters

The design consists of circuit boards that stack one upon another (see Fig 2). They are connected together via "board stacking" connectors (see Figs 3 and 13). This method allows me to design a system that is easy to modify, expand and customize. It allows the builder to implement up to 48 LA channels (in multiples of eight) or a maximum of six analog channels and any combination thereof. The resulting assembly is fairly close to a 4.5-inch cube with two analog cards and one LA card. Great care was taken in the design of the circuit boards in order to achieve this modularity. A template board having just the interboard connectors was designed first. It was then used as the basis for all the boards.

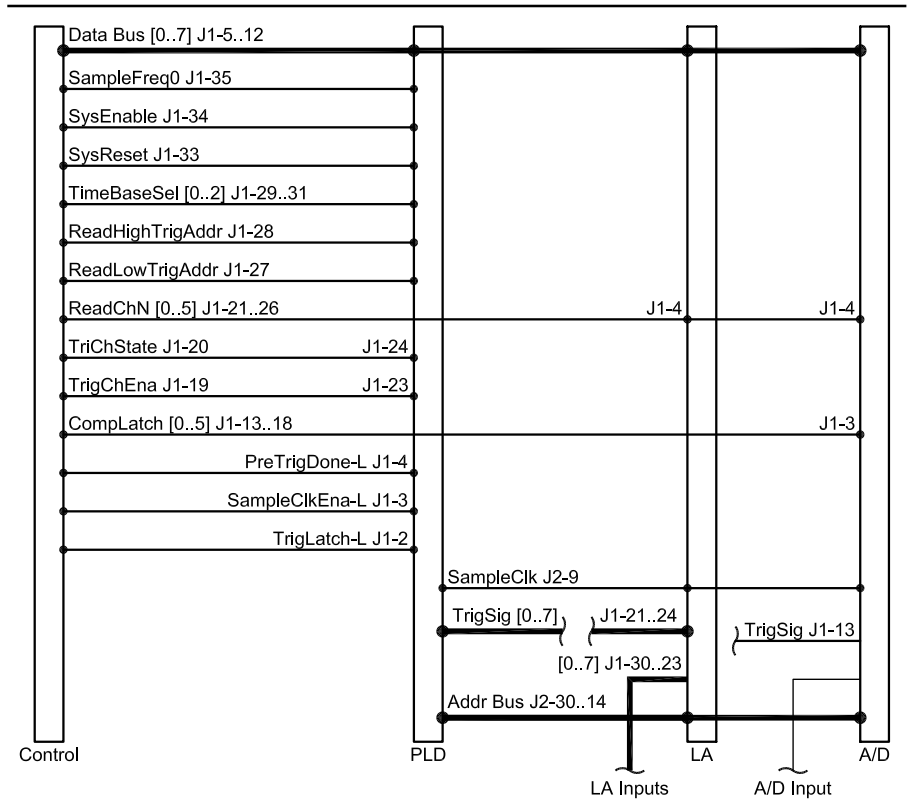

Fig 3—A wiring diagram of the connections between boards.

I wanted to achieve a 20-MHz sample rate so that the analog channels would be useful to 10 MHz. The A/D converter selected (ADS820) is specified to a 20-MHz sample rate, a 60-MHz bandwidth and has 10-bit resolution. Since the circuit uses only the most significant eight bits, some errors of the A/D can be ignored. A higher-frequency A/D IC is available, and it is pin compatible with the unit I selected, but I doubt my ability to implement a circuit board that would allow the additional band-width. The circuit appears in Fig 4.

Major Circuits

- Control—interfaces to the printer port, source for all the static control signals.
- Trigger Address—stores the address at which the trigger occurs.
- Trigger Control—develops most of the dynamic control signals, pre/post trigger count.
- Trigger Detect—detects the trigger condition.
- Address Generator—17-bit address = 132 k.
- Time Base—generates the sample clock frequencies.
- LA—CMOS levels, eight channels, 132 k memory depth.
- A/D Converter—+0.5 to +4.5 V, 8-bit resolution using a 10-bit A/D and 132 k memory depth.

Design History

During the various phases of building and debugging the circuits, I developed several versions of some circuits. Initially, the above circuits were built on seven circuit boards. I managed to implement the Address-Generator and the Trigger-Address circuits on opposite sides of the same board. I used this implementation to get the initial hardware and software design working. The completed assembly was an approximately 4.5-inch cube with one LA and one A/D board. Since I do not have through-hole construction capability, there were quite a few hand-wired jumpers on the various boards.

I then decided to try my hand at using *programmable logic devices* (PLDs). Design number two combined the Time-Base and Address-Generator circuits into a single PLD and the Trigger-Control and Trigger-Detect circuits into another PLD. The PLDs are on opposite sides of the same circuit board. I had never used PLDs before, so this was a learning experience. Getting the software to work with this design was relatively easy, since I was able to keep the design essentially the same, with only some minor changes. This design has five circuit boards.

Design three combines the following circuits into a single PLD: Trigger Address (Fig 5), Trigger Control (Fig 6), Trigger Detect (Fig 7), Address Generator and Time Base (Fig 8). This yields a much more compact system that requires only two circuit boards more than the desired input boards. Additional wiring is shown in Fig 9.[1]

I would like to encourage any of you designing logic circuits to consider PLDs. My experience was most satisfactory. The free software is very easy to use, especially since I had already debugged my initial circuitry. It was only necessary to translate my schematics to functions available in the PLD design software. The "larger" functions I needed were available as library elements. To give you a better idea of the ease of use of PLDs (programmable logic devices), I will expand on the 17-bit counter example.

The address generator requires a 17-bit (131,072 count) synchronous counter. A synchronous counter is required so that all the address bits change at the same time, synchronously. The discrete circuit I initially implemented is shown in Fig 10. Notice that it uses five ICs. Even though I use only one stage of the fifth device, it is required to maintain synchronous operation.

The PLD implementation is quite a bit simpler (see Fig 8). I needed to select the synchronous counter from a list of device models, select and define the needed parameters and place it in my schematic. A complete list of possible parameters appears as Table 2. The only inputs I needed were *aclr* and *clk*, the only output was *q[]* and the parameters I specified were direction and width.

Once the part was placed, I then needed to "wire" the I/O ports to the appropriate circuits or I/O pins. See the AddrClock circuit of the Time Base and Address Generator schematic.

When the schematic entry has been completed, the next step is to "compile" it. This basically assigns the schematic elements to the internal circuit blocks of the PLD and determines whether or not the circuit fits. If the circuit does not fit, there are several options available that can be used to reduce the amount of resources used. Some of these are:

1. Disabling JTAG (Joint Test Action Group) capability.
2. Disabling globally assigned signals such as clocks and clears.

Once this stage of compilation is complete, the pin assignments must be made. This can be done automatically or manually. I always chose manually, because I wanted the PLD I/O to allow the easiest printed circuit board layout. The manual method is really quite easy. I had previously determined the signal-to-pin assignments based on the PC board layout. The PLD software presents a list of signals as well as a "picture" of the PLD. All I had to do was "drag and drop" each signal onto the desired pin.

Now the final compilation phase can be executed. This phase attempts to "fit" the circuit into the PLD with the pin assignments. The software I used does several iterations to complete the fitting process. I have seen up to 20 iterations. If it is successful, a file is created that is used to program the PLD. If the process is not successful, some pin reassignment will be necessary.

Finally, it is time to program the PLD. A programming device or equivalent circuit is required. I was able to obtain a programmer from the PLD manufacturer; but, since the circuit consists of a common IC and a few resistors, it would not be difficult to build one. The PLD I selected is capable of in-system programming. This requires only that I implement the appropriate connector on the circuit board with four connections to specific PLD pins. These four pins can be used for I/O, but I chose to not do that. The programmer connects to the printer port of the PC and the connector on my board. Once the PC software programmed the PLD, I removed the programming cable from the board.

The circuit is now ready for the "smoke" test! Since it is impossible to probe the internal circuit points, I strongly suggest that test points be designed into the PLD circuit. At one point of my design cycle, I had three of them. These allowed me to 'scope some of the critical timing points. Because I had previously implemented identical circuits using discrete logic, my debugging was minimal, so I did not need many test points. Untested circuits will probably require more. I

Table 1—Printer Port Signal Usage

Output Control Bits

Pin	*Printer Port Name*	*Usage*
1	/C0: Strobe	RegSel 0
14	/C1: Auto-Linefeed	RegSel 1
16	C2: Initialize	RegSel 2
17	/C3: Select-Printer	Strobe

Input Control Bits

Pin	*Printer Port Name*	*Usage*
10	S6: Ack	PreTrigDone-L
11	/S7: Busy	
12	S5: Paper-Out	AddrClockEna-L
13	S4: Select	Trigger Latch-L
15	S3: Error	

Data Bits

Pin	*Printer Port Name*	*Usage*
2 - 9	D0 - D7	Data
18	Ground	

[1]A package containing additional details of the PLD connections and PC-board etching patterns is available from *ARRLWeb.* You can download this package at **http://www.arrl.org/qexfiles/**. Look for 0307CICC.ZIP.

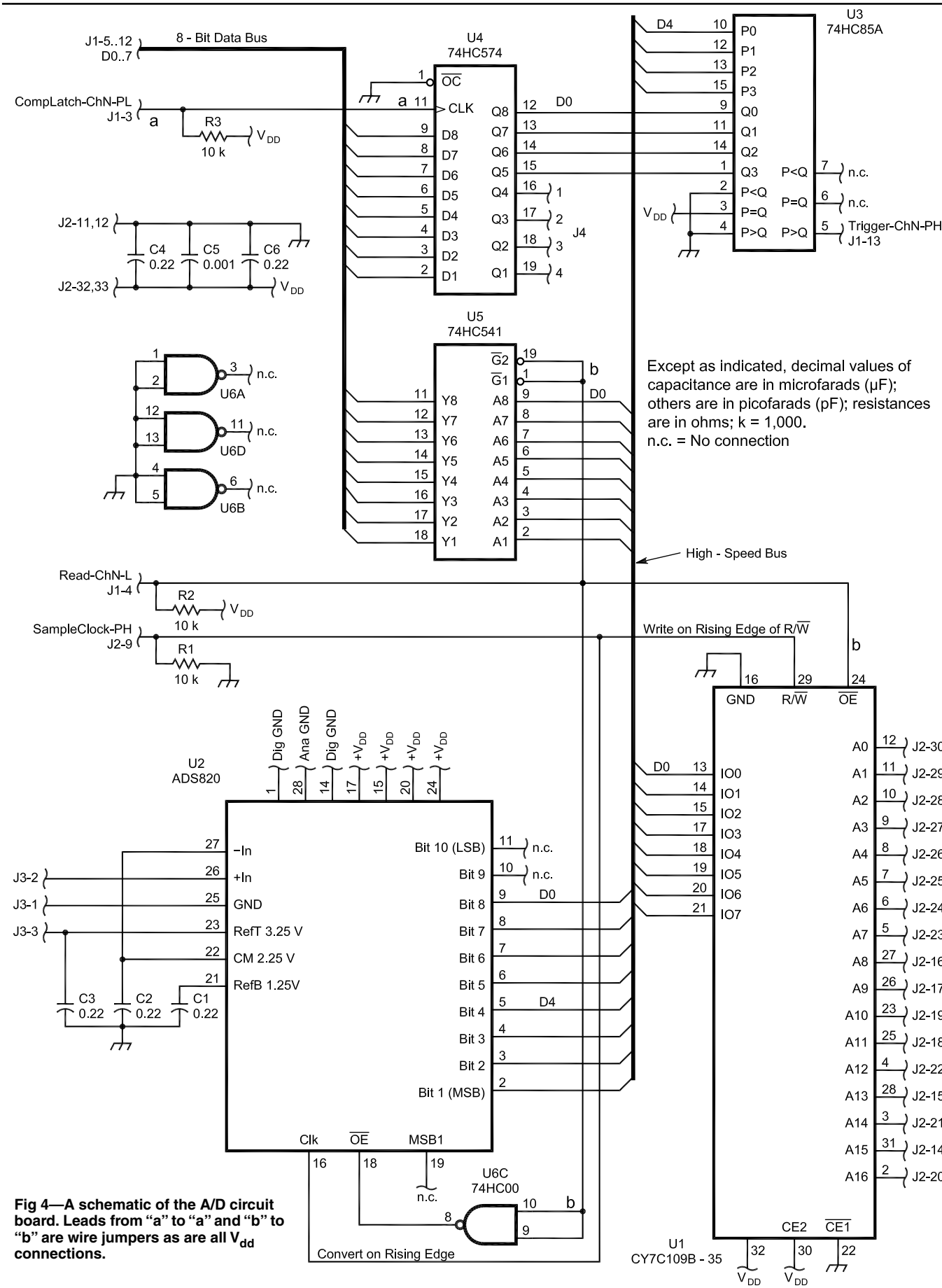

Fig 4—A schematic of the A/D circuit board. Leads from "a" to "a" and "b" to "b" are wire jumpers as are all V_{dd} connections.

recommend that outputs be buffered so that external loading of the I/O pins does not adversely affect system operation.

This was my first attempt at using PLDs, and I found it to be much easier than expected. The free software, downloaded from the Web, is fairly intuitive and comes with a tutorial. I ran the tutorial through the first dozen or so steps; then, like a typical engineer and ham, went off on my own. I was able to go from starting with no previous experience to a finished product within a few weeks working an hour or so several evenings a week. Table 2 shows a complete list of the possible I/O signals and parameters that may be used to define the operation of the counter; Table 3 lists the parts used in each board.

Circuit Descriptions

I have followed a naming convention for the signal names to more easily determine their operation and make the schematics and other documentation easier to follow:

1. Dynamic signals have a "PL" or "PH" suffix, indicating "pulse low" or "pulse high."
   Example: *SysReset-PL*.
2. Static signals have an "H" or "L" suffix indicating a "High" true or "Low" true state.
   Example: *Read-Ch0-L*.
3. Static signals, which are neither high nor low true, do not have a suffix.
   Example: *TimeBaseSel0*

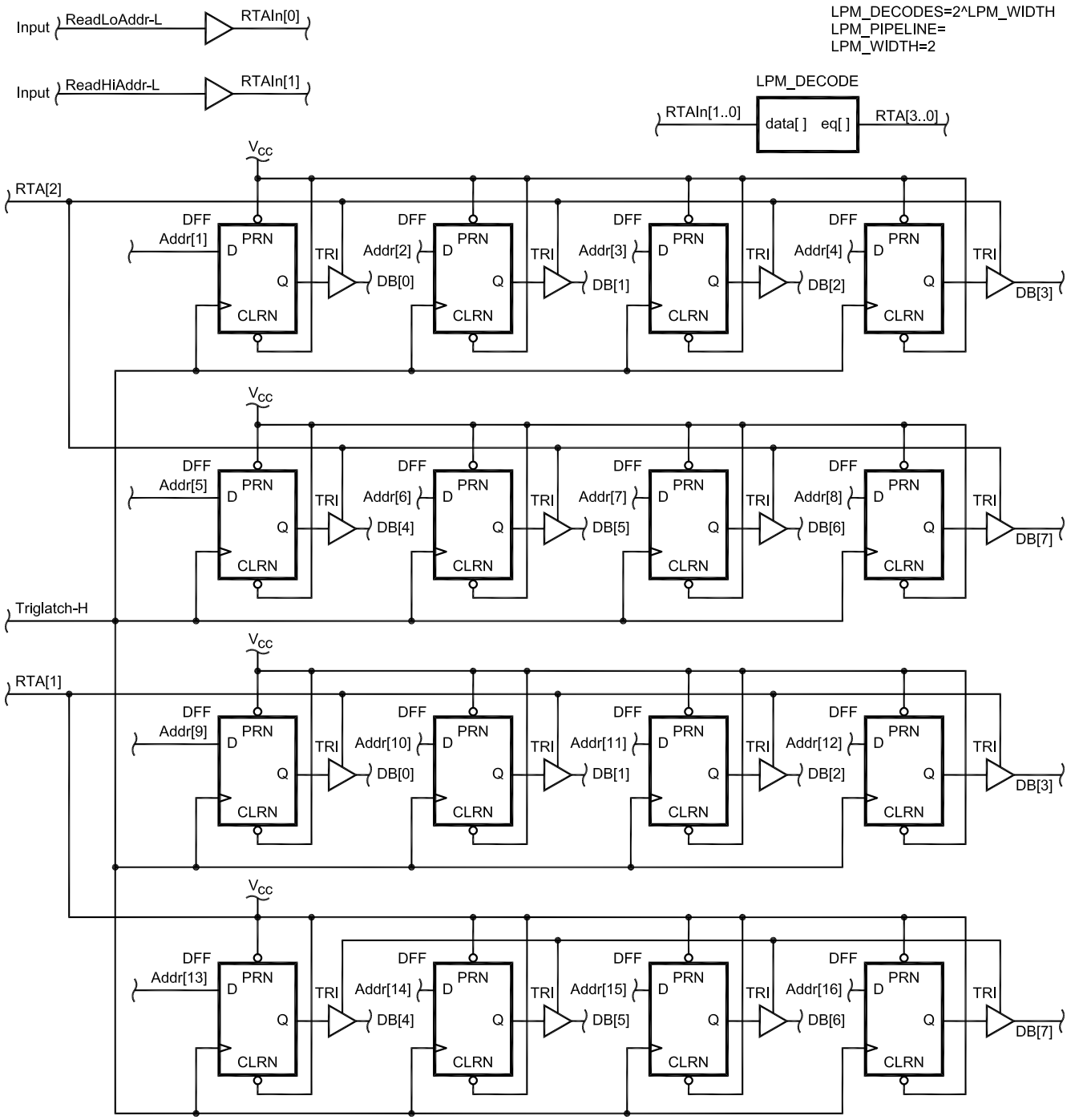

Fig 5—A schematic of the PLD Trigger Address Latch circuit.

In general, I have labeled only signals that go between circuits.

Control Circuit

The main purposes of the Control Circuit (Fig 12 and 13) are to provide the interface to the PC (parallel port) and develop the static signals used to control the remaining circuits. The printer-port control bits are used to control the primary-decoder (U1) function. The functions performed by the primary decoder are:

- System Reset
- System Enable
- Secondary Decoder and Time Base Select Strobe—U2
- Read Clock and Secondary Decoder Read Strobe—U5
- Secondary Decoder Write Strobe—U4

Except for the Time Base Select signals, all of the signals generated by this circuit are low-true pulses.

There is also a bus transceiver (U3) that controls the direction of data flow between the PC and the DSO/LA. It uses the *ReadClock* to determine the data direction. When *ReadClock* is low, read from the DSO/LA; when it's high, write to DSO/LA. Although many of the circuits allow for customization, this one does not, since the software is written to correspond with the functions it performs.

Time Base

The operation of the Time Base is rather straightforward (see Fig 8). There are three control bits (with possible expansion to four) that come from the Control Circuit (*TimeBaseSel*). These bits control a multiplexer that is used to select from among eight signals that can clock the Address Generator (*AddrClk-PH*). Seven of these are also the Sample Clock signal for both the A/D and LA circuits (*SampleClock-PH*). The eighth signal is for advancing the Address Generator when the program needs to read the data from the A/D and LA RAMs. The 20-MHz oscillator and Sample Clock are enabled by a signal from the Trigger Control Circuit (*SampleClockEna-H*).

Notice that the signal driving the Address Generator goes through three inverters, while the signal that drives the Sample Clock only goes through a single gate. There are two reasons for this configuration:

1. The Sample Clock should be disabled when reading from the A/D and LA Circuits.
2. The Address Clock is delayed

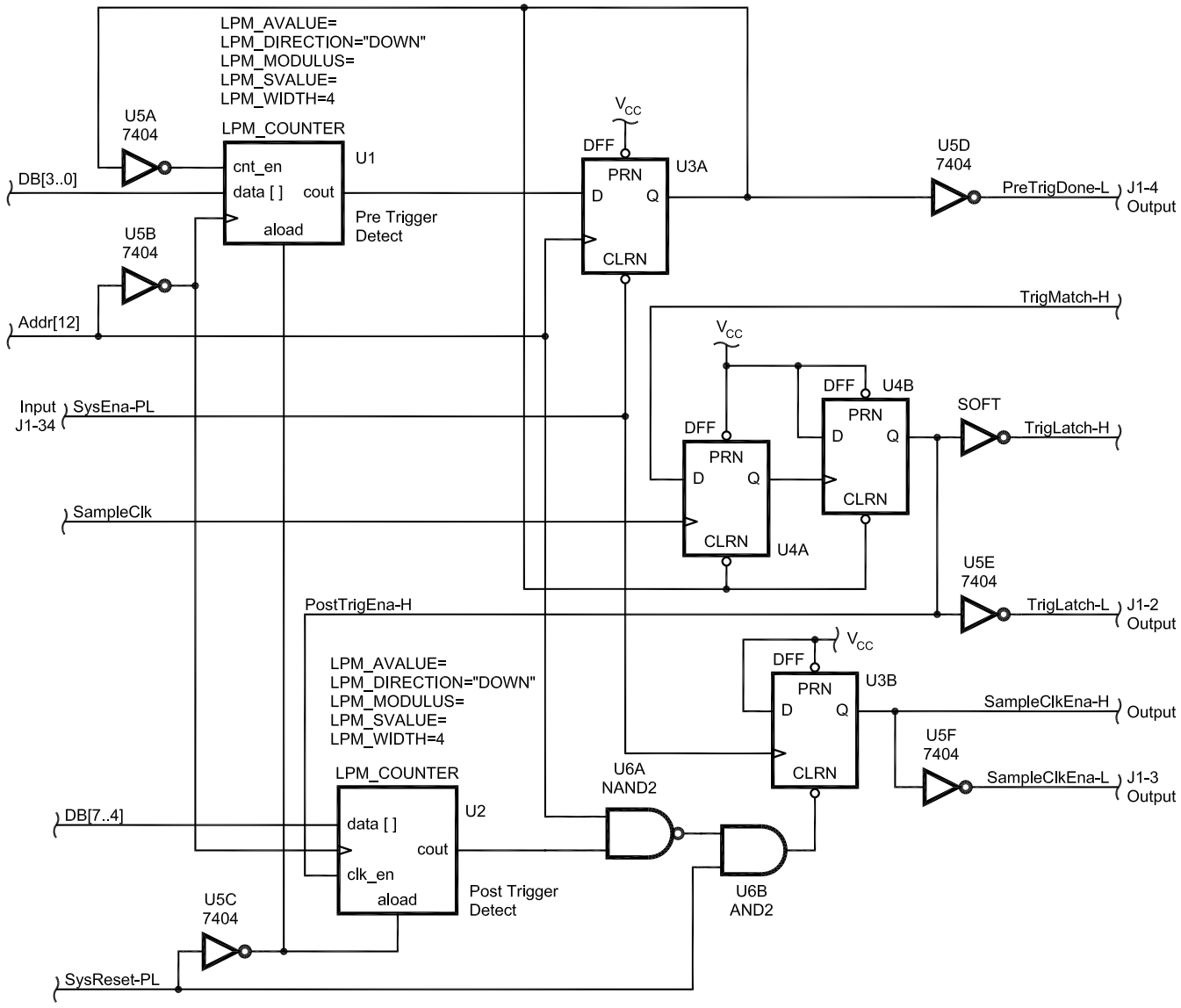

Fig 6—A schematic of the PLD Trigger Control circuit.

slightly so that the input signals are sampled a short time (two gate delays) before the address bus is incremented. This allows the input signals and address bus to settle for almost a full clock period before the data is sampled and is really only significant at the highest sample rate but is necessary to meet the access time specification of the RAM.

I elected to make the sample rate/period circuit as simple as possible. I have used binary dividers to generate the various sample rates. Starting with a 20-MHz clock the next rate is 10 MHz, followed by 5 MHz and so on. The sample periods with this circuit are: 50 ns, 100 ns, 200 ns, 400 ns, 800 ns, 1.6 µs and 3.2 µs. These may seem like strange values, but it really does not make any practical difference in the interpretation of the displayed signals. Since the software allows you to measure the time between cursors, it accounts for the sample period. Also, the software allows you to set the time scale.

The design of the system is flexible, so if you prefer to use sample periods different from what I have chosen, the circuit can be redesigned to implement the ones you like. I will be more than happy to customize the PLD for anyone building this system. The sample periods must be defined in the INI file. Simply relate the decoder input value to the sample period as follows: `SELECT_<decode value:1..15> = <sample period>`. For example: `SELECT_1 = 50 ns`. The program will correctly interpret nanoseconds, microseconds and milliseconds as units of time.

Address Generator

This is a 17-bit synchronous binary counter that yields 2^{17} = 131,072 addresses (see Fig 8). It is driven by the Address Clock from the Time Base Circuit. The counter outputs are all set to zero by the *SysReset-PL* signal from the Control Circuit. Because of the way the Pre and Post Trigger circuits operate, the full count is not utilized. This will be explained further in the discussion of the Trigger Control circuit.

The original design of the circuit called for 74HC161 binary counters that are specified to 25 MHz with a 5-V supply. When more than two units are cascaded, however, there is a glitch. I had not thoroughly read the details of the specification and missed the information on the glitch. I thought that since they were rated to 25 MHz, I would not have a problem since I was only going up to 20 MHz. The glitch occurs because of internal delays when using the Ripple Carry Out. This limits the upper frequency to 17 MHz when cascading. The solution was relatively simple: use 74AC161 devices instead. They are rated at a much higher frequency but draw quite a bit more current. Since I have now converted the circuit to a PLD, the above situation is no longer applicable, but I thought it worth mentioning since it was part of my learning experience.

Trigger Detect

Those of you who have used commercial LAs will recognize that the triggering available with this design is very limited. These units will usually have quite a few possibilities

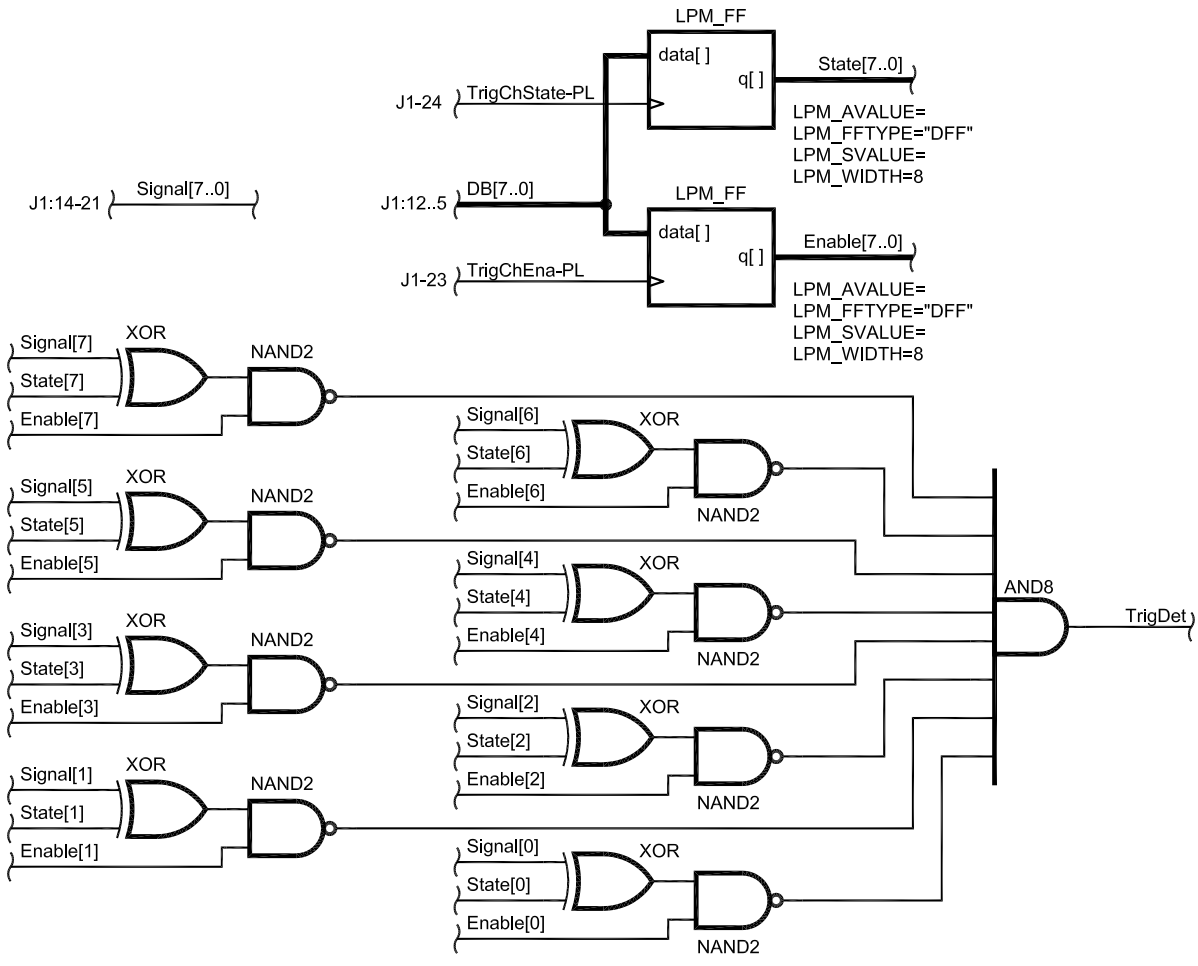

Fig 7—A schematic of the PLD Trigger Detect circuit.

for triggering such as:

- High level
- Low level
- Low to high transition
- High to low transition
- Either transition
- Various logic combinations of selected signals

My circuit implements only these three:

- High level
- Low level
- Ignore the signal

This circuit compares the current state of the trigger inputs to those that have been selected by the operator. The circuit operates as an 8-bit **AND** gate (see Fig 7). Each of the eight input signals can be enabled or disabled and inverted or not inverted. The source for each of the eight inputs is up to you. I have mine set up so that the first six signals are channels 0 through 5 of the first LA board. The remaining two inputs are from the two A/D boards. This is one area where the system is not very flexible. You cannot dynamically select from among all the possible inputs. You must predetermine which eight signals you will use as possible triggers. If this is not flexible enough for your applications, you will need to implement a switch of some kind. I am certainly open to suggestions about how to make this function more flexible.

Each exclusive-or gate is used as programmable inverter. When its control input is low, the output follows the input. When the control input is high,

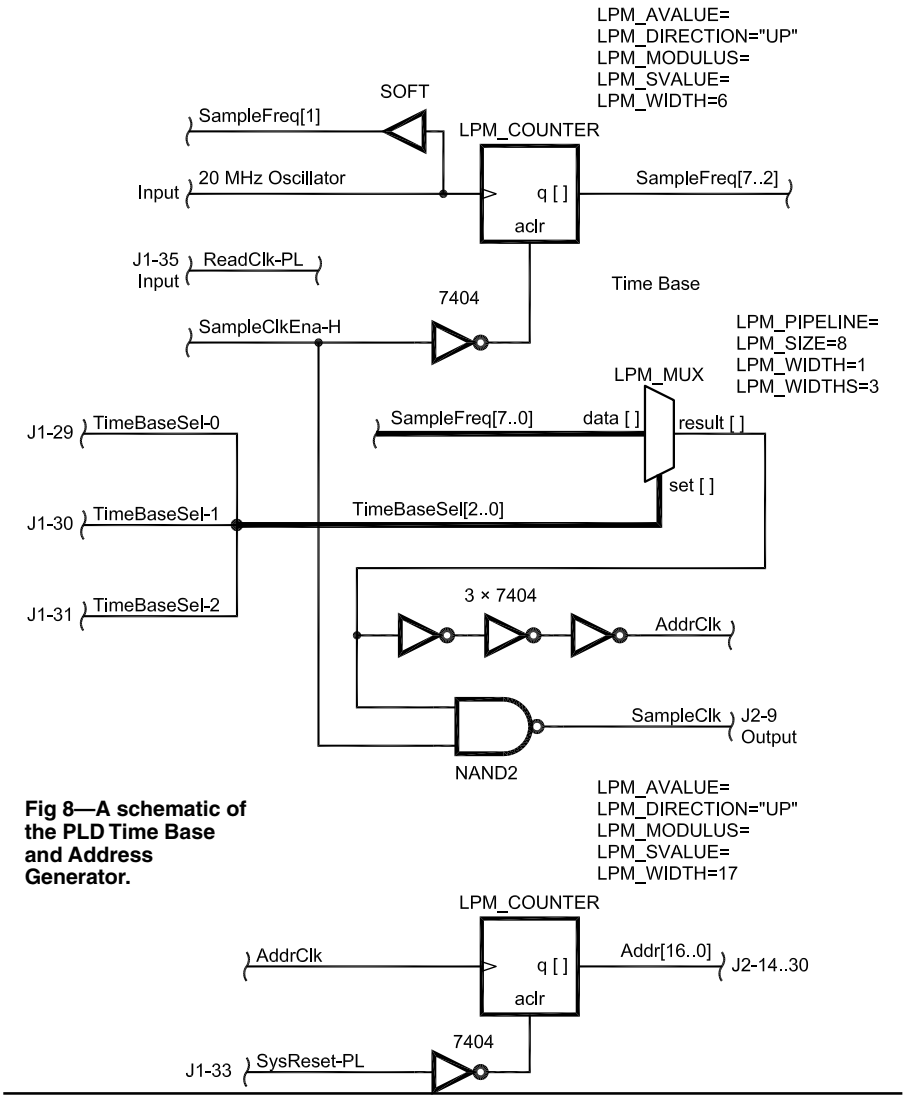

Fig 8—A schematic of the PLD Time Base and Address Generator.

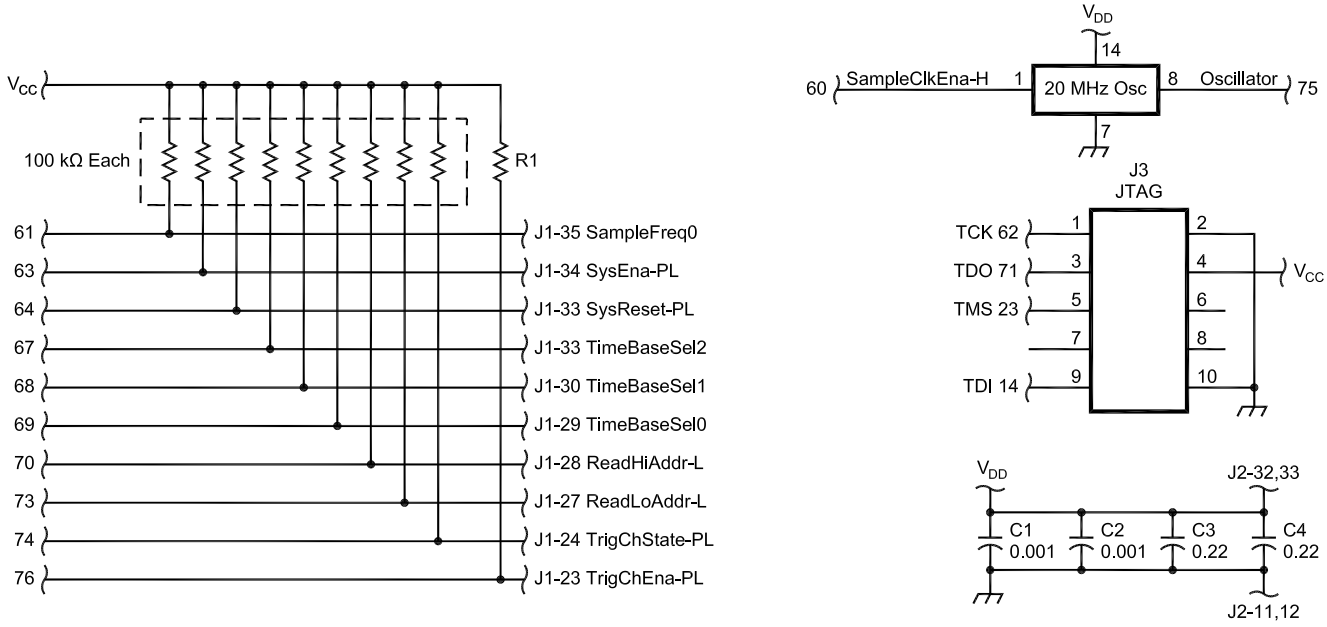

Fig 9—A schematic of the PLD board. Numbers without connector designations (J1 or J2) are on the PLD (EPM7128SLC84). See Table 2 for most signal connections.

the output will be the inversion of the input. The following **AND** gate is used to enable/disable the input signal. When its control signal is low, the output is forced high, independent of the input signal. This basically creates a match condition for the final eight-input **AND** gate. If all the signal inputs are disabled, there will be a continuous match condition.

The original, discrete-logic version of the circuit was quite different. The 8-bit latches were 74HC574s, very similar to the PLD implementation. However, the enable/disable circuit used tri-state buffers (74HC126) and the output **AND** gate was actually an 8-bit comparator (74HC688). The circuit required five ICs. If I were to implement the circuit I designed for the PLD as discrete ICs, it would require seven. Since the circuit is now in a PLD, I can probably modify it to meet some other requirements.

Trigger Control

The Trigger Control (Fig 6) was perhaps the most difficult circuit to design and get working as I wanted. My goal was to have a programmable pre-trigger capability—unlike my original LA, which has a fixed amount of pre-trigger. I have added "U" numbers to the schematic of the PLD circuit to make it easier to reference the different parts of the circuit. Both U1 (the pre-trigger counter) and U2 (the post-trigger counter) are 4-bit down counters. They are clocked by the inverted A12 signal. As such, they decrement every 8192 samples.

You can set the pre-trigger count to any value between 0 and 14. The number of pre-trigger samples can be determined by this formula: pre-trigger samples = (pre-trigger count × 8192) + 4096. The post-trigger count is set by the system as follows: post-trigger count = 15 – pre-trigger count. The total number of samples is $(15 \times 8192) + 4096 = 126976 = 2^{17} - 4096$.

Once the system is "armed," the pre-trigger counter starts counting down to 0. When it reaches 0, the output of U3A will go high on the next positive transition of A12 (4096 samples later). This enables a Trigger Match to be detected. A Trigger Match will then cause the *Q* outputs of U4A and U4B to go high, thus enabling the post-trigger counter. When it reaches 0, the sample clock will be disabled and the system will stop sampling.

Three status signals are monitored by the program that allow it to determine the state of the system:

- *PreTrigDone-L*
- *TrigLatch-L*
- *SampleClockEna-L*

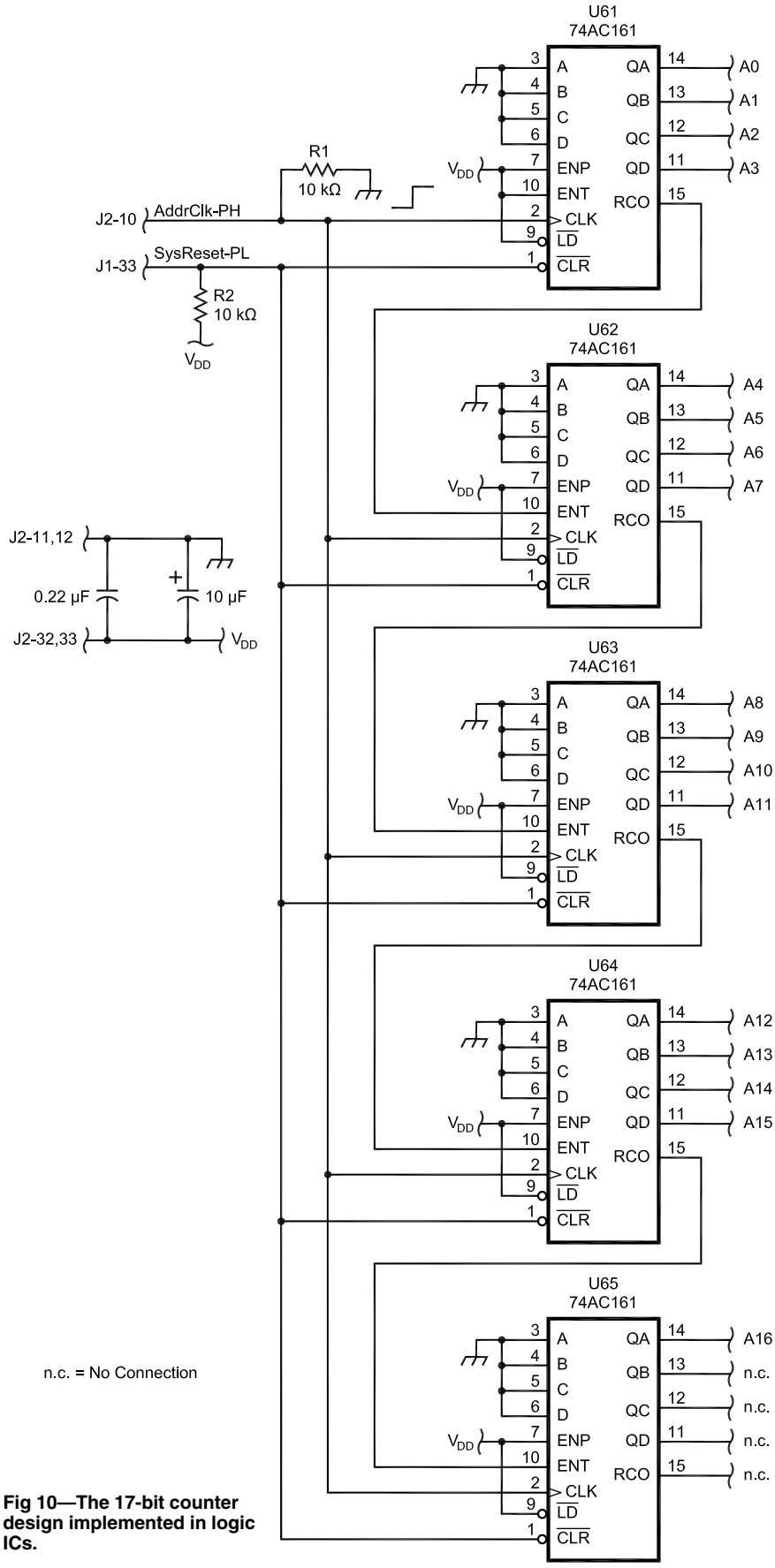

Fig 10—The 17-bit counter design implemented in logic ICs.

A message is displayed as the system waits for each of these signals. Please notice that each of these signals is derived from signals used internally by the Trigger Circuit. However, I have buffered them via sections of U5 so that the wiring associated with getting them to the PC does not interfere with the circuit operation.

The PLD version of the circuit is almost identical to the discrete version. I would like to detail some of the differences for those of you who wish to convert a circuit to a PLD.

The carry out of PLD counters is true when high and becomes active as soon as the count reaches 0 or 15, de-

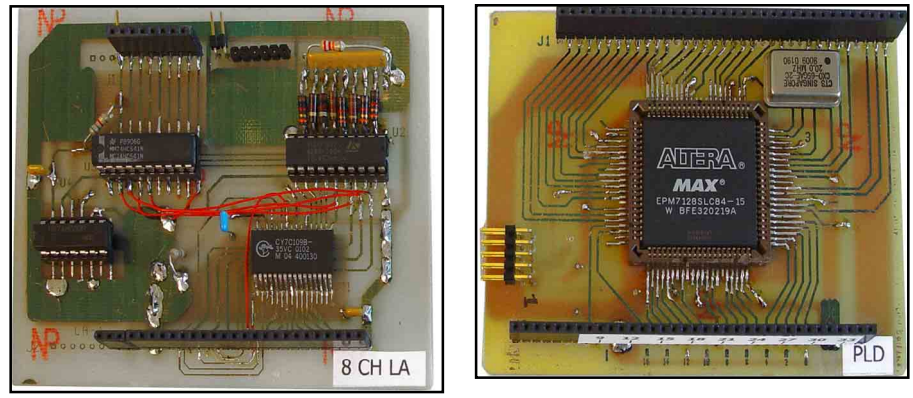

Fig 11—The 8-channel LA board and the PLD board. Notice the two board-stacking connectors on the PLD board.

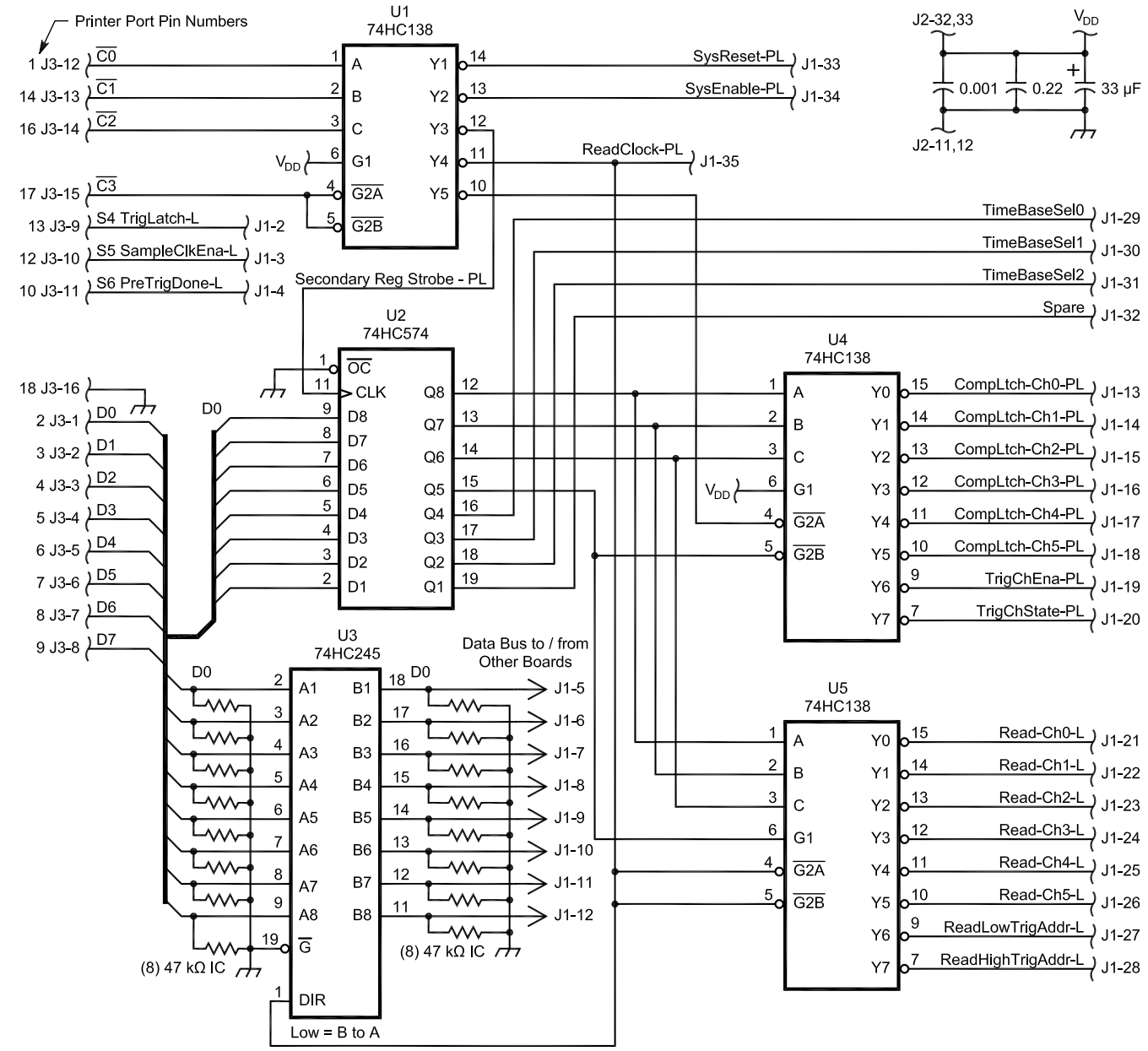

Fig 12—A schematic of the DSO/LA control board.

pending on the count direction. The discrete device I used is a 74HC191. Its carry out is true when low and is gated with the clock so that it goes true during the second half of the end count.

Both counter-control inputs, Count Enable and Load, are true when high for the PLD but are true when low for the '191.

The discrete version required an **AND** function so I used a 74HC00 as a **NAND** and then followed it with another section to invert it. This is certainly not necessary in a PLD, since it has built-in **AND** blocks.

LA

This is another relatively simple circuit (see Fig 14). All it must do is apply eight input signals to the data bus of the RAM and allow the data bus to be read back by the PC. There are two resistor networks in the input circuit. One insures that the input-switch device has no floating inputs. The other limits current in case of excessive applied signal voltage. As can be seen on the schematic, U2 and U3 create an eight-pole, two-position switch controlled by the *Read-ChN-L* signal. When *Read-ChN-L* is high, the RAM data bus is connected to the input signals, and the *SampleClock-PH* signal writes the data to the RAM on its rising edges. The *SampleClock* signal is high during the read operation. When *Read-ChN-L* is low, the RAM data bus is connected to the system data bus, which is eventually read by the PC. Each LA board has its own *Read-ChN-L* from the Control circuit.

Please note that the address bits driving the RAM are "scrambled." This was done in order to make the circuit-board layout easier. It was much easier to route the address signals to the RAM from J2 using this technique as opposed to routing the address lines to their "correct" pins. This is a common technique in industry. The RAM really does not care which signal from the address generator is connected to which address input. I have seen many RAM devices that do not assign specific address bits to their address inputs.

A/D

The circuit is somewhat similar in operation to the LA except that the eight data bits come from an A/D channel instead of individual digital sources (see Fig 4). The *ReadChN-L* signal is used to determine whether the circuit is sampling A/D data or transferring it to the PC. U5 is used to connect the RAM data to the system data bus when *ReadChN-L* is low. When this signal is high it allows the *SampleClock-PH* signal to clock the A/D and write data to the RAM.

The A/D I selected (ADS820) requires a single 5-V power supply and has a maximum 20-MHz sample rate and a full power bandwidth of 65 MHz. There are faster pin-compatible versions available, but I did not feel that I could adequately implement the circuits for higher frequencies. There is nothing in the system design, however, that would prohibit you from implementing a faster A/D if you desire. The RAM (CY7C109B-35) is good up to about 28 MHz but has faster versions. The -20 is a 50-MHz part.

When sampling the input signal, the rising edge of *SampleClock-PH* initiates an A/D conversion and writes A/D data to the RAM. These types of A/D converters have a pipeline architecture. In the ADS820, this causes a six-cycle latency between when a signal is sampled and when the converted data is available on its output bus. In my design, this means that I need to account for the latency in the software so that I store the data in the correct place.

There are two additional functions on this card:

- Digital comparator
- Four-bit latch for controlling the A/D buffer

The digital comparator allows the user to set a trigger level that can then be used to trigger the system. On the schematic, notice that I have labeled the RAM data bus as "High Speed Bus." I tried to be as careful as possible in routing these signals in order to maintain signal integrity. The comparator is operating on the A/D samples at the sample rate, as high as 20 MHz. When the digitized signal value is greater than the comparison value stored in the upper four bits of U4, the **P>Q** output of U3 will go high. This signal can be used to trigger the system. I have implemented a four-bit comparator so the resolution is only 1 part in 16 (6.25%). I felt this was a reasonable compromise so that I could use the remaining four bits to control the A/D buffer.

The timing requirements for writing

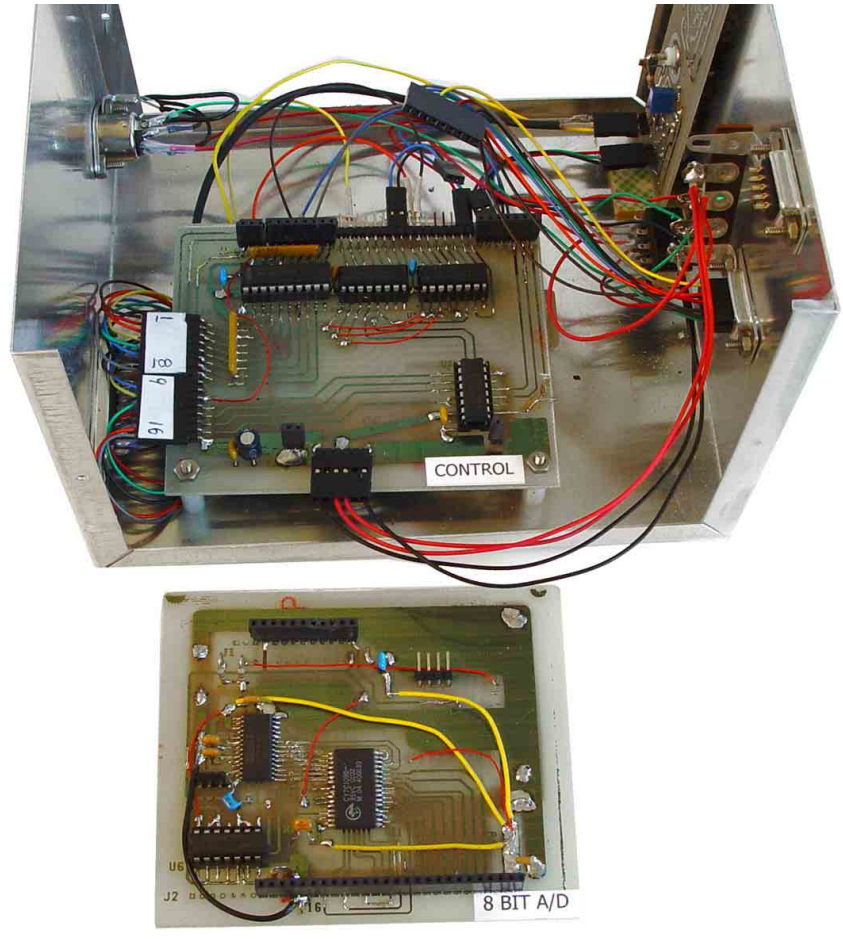

Fig 13—A photo of the Control circuit board and one 8-bit A/D board. Two of the five jumpers are there to "fix" broken traces. The other three are designed-in because I could not route them. There is one IC mounted on the underside.

A/D data to the RAM can cause problems if the circuit design is not carefully considered. A careful inspection of the A/D timing shows a "dead" time on its data bus immediately following the positive transition of the clock. Note that writing to the RAM also occurs on the rising edge. At first I thought this might cause a problem. However, the A/D specifications show that the data is good for a minimum of 3.9 ns after the rising edge, while the RAM specifications show a 0 ns input hold time for the data. Although 3.9 ns does not seem like very much, it is sufficient for the circuit to operate properly.

A/D Buffer

This circuit (see Fig 15) is built on a board that is not part of the "stack" forming the main body of the instrument. I wanted it as close as possible to the signal source, so I designed a small board that mounts directly to the front of the LA/DSO housing and has the input (BNC) connector mounted to the board. This keeps the shortest possible signal path to the buffering op-amp. This is a simple dual op-amp circuit that has three basic functions:

1. Buffer the signal, which may be up to 10 MHz.
2. Input attenuator to allow a –5 to +5 V input span.
3. Condition the signal to the 4-V span of the A/D converter.
4. Offset adjustment to center the signal about 2.5 V.

This particular op amp has relatively high input and offset voltages and currents so its dc characteristics are not the best. Nonetheless, it is fast enough to handle 10-MHz signals as long as you do not require the full 8-bit accuracy. There are op amps

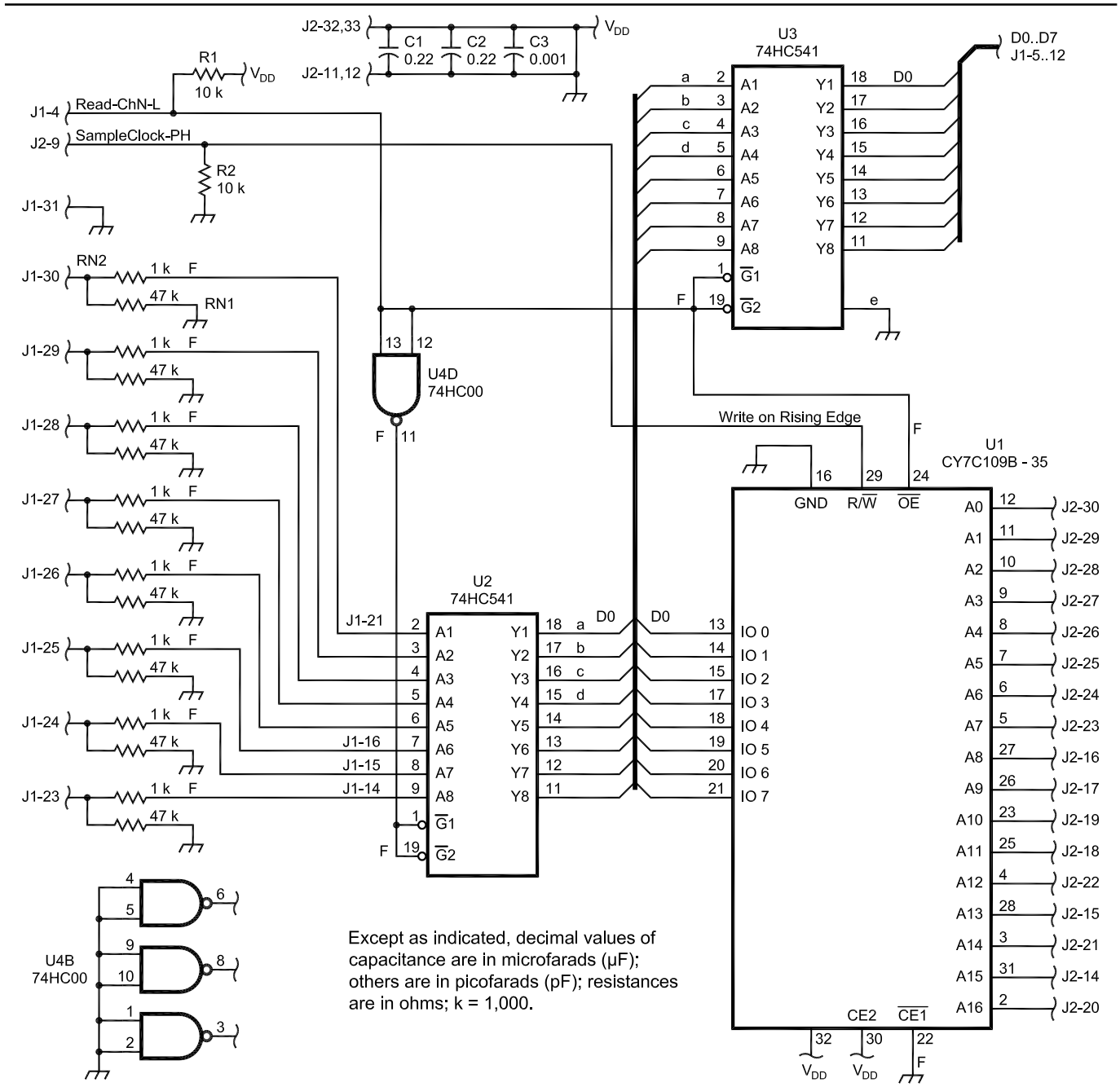

Fig 14—A schematic of a logic-analyzer circuit board. Wires labeled "a," "b," "c," "d" and "e" are jumpers. Letters "F" indicate a connection between the two copper layers of the circuit board.

available that would be better, but they are probably more expensive and this great expense was not one of my primary design goals.

The circuit I designed does not use the four control bits available from the A/D card. I did not want to add this complexity to my circuit for the first design. This means that the basic input level is –5 to +5 V. I use attenuator probes to extend the range as required. I will be designing a programmable attenuator board sometime in the near future and will post it on my Web site (**www.qsl.net/k3pto/**). The input ranges are completely flexible, and you can make them whatever you want. The INI file allows you to specify the ranges associated with the four control bits (16 values) coming from the A/D circuit.

Trigger Address

This is perhaps the simplest of the circuits. It is simply a 16-bit latch that is triggered by the output of the Trigger Detect Circuit via the Trigger Control Circuit. This was done so that only a single trigger event will cause the address to be latched. I used a 16-bit latch, implemented via two 8-bit latches, because I did not want to build the necessary additional circuitry for the 17th bit. The program will read the upper 16 bits and then determine if the data actually meets the trigger criteria. If it does not, then it must be the next higher address. This little bit of extra programming saved me several ICs in the circuit design. However, now that I am using a PLD, I may go back and revisit this as I work on improving the system.

System Operation

The operation of the DSO/LA is fairly intuitive. You can get the basics simply by seeing the pictures that follow. There will be a complete Help file available, which explains each of the controls as well as the contents of the INI file. Here is a brief explanation of each of the menu items:

- File—typical file operations for reading and writing both INI and data files.
- Boards—define the board for each of the six channels.
- Trigger—define trigger conditions, including the pre-trigger/post-trigger values.
- Display—select Time Base, Zoom factor, Slide factor and so on.
- Cursors—enable the four time-measurement cursors.
- Function Keys—list the applicable function keys.
- Debug—some hardware-debugging aids as well as some dummy data so you can "play" with the display.

Future Enhancements

I am in the process of redesigning the layout of both the A/D and Logic Analyzer boards to make them easier to build. Both will probably have a PLD to replace the logic devices. This will also make it easier to lay out. These new boards will be totally backward compatible with the current designs so that they can be installed in my current system.

As indicated at the beginning of the article, I am considering a serial interface. If I design one it will probably have Ethernet capability also. I have not completely thought this through yet, but with an Ethernet interface it is easy to operate the unit over the Internet.

Once the data has been captured, it can be analyzed in many ways. Currently, I simply display it. I plan to add several measurement capabilities by the time this article is printed. Some possibilities are:

1. Count the number of logic transitions between cursors.
2. Decode an ASCII bit stream.
3. Decode SPI—synchronous data with a clock.
4. Decode I²C—more difficult than SPI but still a possibility.
5. Perform some limited math operations on both the LA and A/D data.
6. Possibly even a limited spectrum analysis—if I can develop or obtain some code.

Suggestions from anyone who builds a system are welcome.

Some Suggestions for Making PC Boards

Make the traces as wide as possible. Most of the boards started out with 20-mil traces. After completing the design I went back and made each trace as wide as possible. If necessary, I narrowed the trace where it passes between IC or connector pins. The undercutting of traces during etching has minimal effect on wide traces.

Make the ground traces even wider to minimize the resistance of ground leads. I routed signal traces and grounds first, ignoring the +V leads. Since this was my first effort at double-sided printed circuit boards and I do not have the ability to implement plated-through holes, I felt it better to use wire jumpers for the +V leads than for signals. This allowed me to use relatively heavy wires for the power leads.

Implement ground fill where possible. This reduces the amount of copper that must be etched as well as giving more ground area.

Use a scrap piece of perforated board as a drilling template for the connector holes. I had some unused circuit boards from RadioShack with 0.1-inch-spaced holes. I lined up the board holes with holes to be drilled and taped the perf board to my circuit board. Viewing the boards with a 50-W lamp behind them made this relatively easy.

I have no drill press, but an attachment for my electric drill allows it to operate as a drill press. Without this

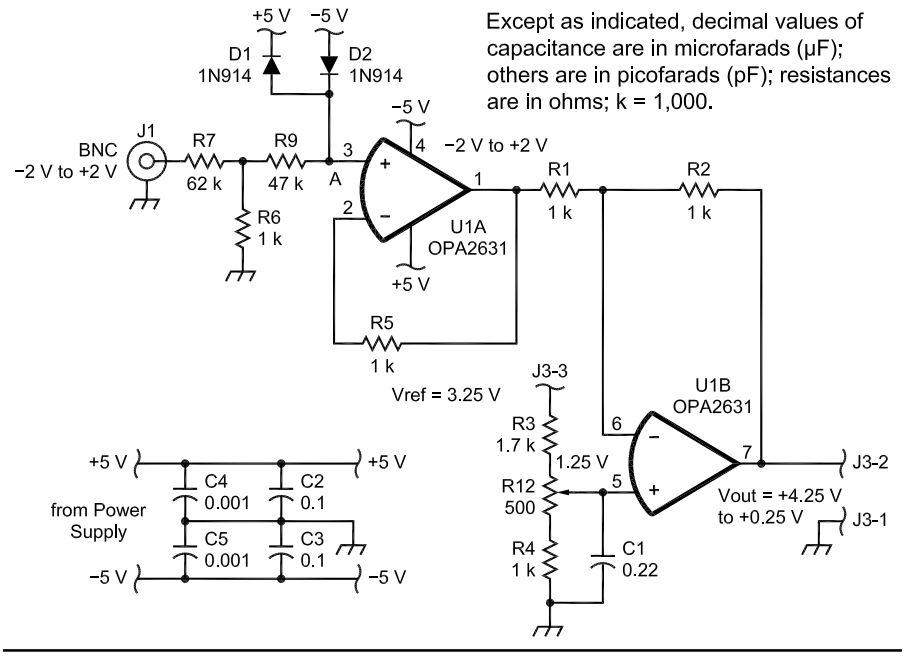

Fig 15—A schematic of the A/D input buffer and conditioning circuits.

device I could not have drilled the circuit boards.

I had a relatively bad experience with drill bits with thickened shanks. The ones I purchased broke rather easily. In order to use the small bits in my drill I wrap thin packaging tape around the upper part of the shank. This worked quite well as long as I wrap the tape carefully.

For those holes that are not on 0.1-inch centers I used a center punch to mark the hole. I modified my punch by cutting off some of the spring so that it makes a smaller dent in the circuit-board material.

Check every connection for both continuity and soldering integrity. I was able to minimize troubleshooting by finding several assembly problems before installing ICs in their sockets and applying power. I also tested every printed wiring trace for continuity after installing all components except ICs.

If the etchant instructions suggest etching at an elevated temperature,

Table 2—Possible I/O Signals and Parameters to define the Operation of the PLD Counter

INPUTS

Port Name	*Required*	*Description / Comments*
data[]	No	Parallel data input to the counter. Input port LPM_WIDTH wide. Uses load or sload.
clock	Yes	Positive-edge-triggered Clock.
clk_en	No	Clock Enable input. Enables all synchronous activities. Default = 1 (enabled).
cnt_en	No	Count Enable input. Disables the count when low (0) without affecting sload, set, or sclr. Default = 1 (enabled).
updown	No	Controls the direction of the count. High (1) = count up. Low (0) = count down. Default = up (1). If the LPM_DIRECTION parameter is used, the updown port cannot be connected. If LPM_DIRECTION is not used, the updown port is optional.
cin	No	Carry-in to the low-order bit. If omitted, the default is 0.
aclr	No	Asynchronous Clear input. Default = 0 (disabled). If both aset and aclr are used and asserted, aclr overrides aset.
aset	No	Asynchronous set input. Default = 0 (disabled). Sets q[] outputs to all 1's, or to the value specified by LPM_AVALUE. If both aset and aclr are used and asserted, aclr overrides aset.
aload	No	Asynchronous load input. Asynchronously loads the counter with the value on the data input. Default = 0 (disabled). If aload is used, data[] must be connected.
sclr	No	Synchronous Clear input. Clears the counter on the next active Clock edge. Default = 0 (disabled). If both sset and sclr are used and asserted, sclr overrides sset.
sset	No	Synchronous set input. Sets the counter on the next active Clock edge. Default = 0 (disabled). Sets q outputs to all 1's, or to the LPM_SVALUE value. If both sset and sclr are used and asserted, sclr overrides sset.
sload	No	Synchronous load input. Loads the counter with data[] on the next active Clock edge. Default = 0 (disabled). If sload is used, data[] must be connected.

OUTPUTS

Port Name	*Required*	*Description / Comments*
q[]	No	Data output from the counter. Output port LPM_WIDTH wide. Either q[] or at least one of the eq[15..0] ports must be connected.
eq[15..0]	No	Counter decode output Active high when the counter reaches the specified count value (AHDL only). Either the q[] port or eq[] port must be connected. Up to c eq ports can be used (0 <= c <= 15). Only the 16 lowest count values are decoded. When the count value is c, the, the eqc output is set high (1). For example, when the count is 0, eq0 = 1, when the count is 1, eq1 = 1. Decoded output for count values of 16 or greater require external decoding. The eq[15..0] outputs are asynchronous to the q[] output.
cout	No	Carry-out (borrow-in) of the MSB.

Parameters

Parameter	*Type*	*Required*	*Description*
LPM_WIDTH	Integer	Yes	The number of bits in the count, or the width of the q[] and data[] ports, if they are used.
LPM_DIRECTION	String	No	Values are "UP," "DOWN," and "UNUSED." If the LPM_DIRECTION parameter is used, the updown port cannot be connected. When the updown port is not connected, the default for the LPM_DIRECTION parameter is "UP."
LPM_MODULUS	Integer	No	The maximum count, plus one. Number of unique states in the counter's cycle. If the load value is larger than the LPM_MODULUS parameter, the behavior of the counter is not specified. This information is from the MAX+plus II help file. The software is available from the Atmel Web site: **www.Atmel.com**.

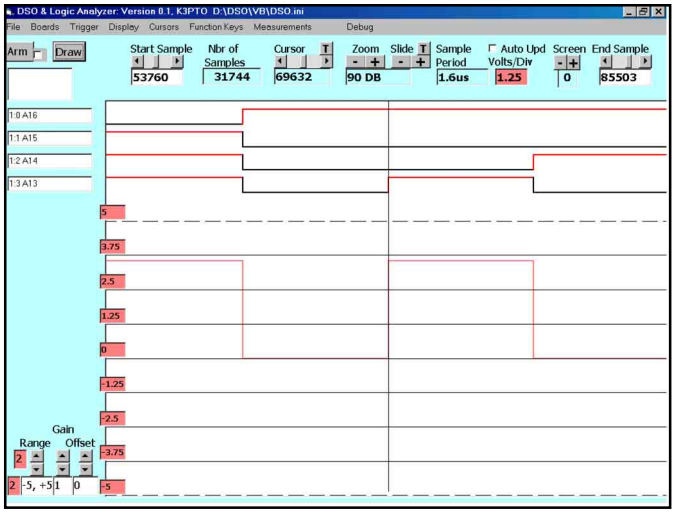

Fig 16—Four address bus signals as captured by an LA board as well as the A12 signal as captured by an A/D board.

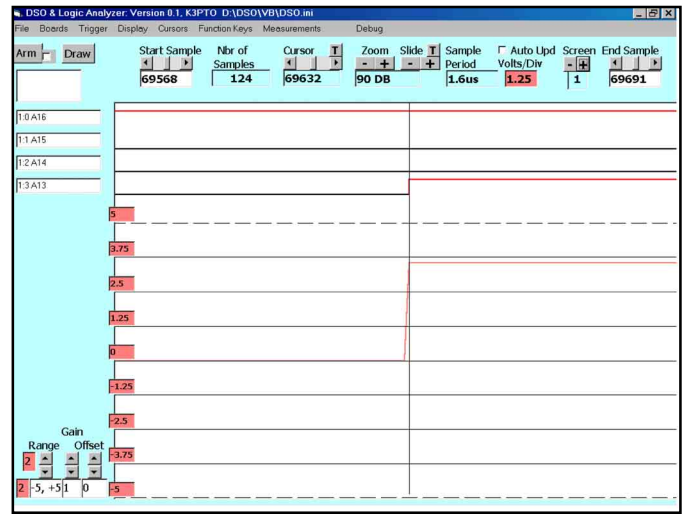

Fig 17—The same signals as Fig 16 but zoomed in.

believe them! The etchant I used, Sodium Persulfate, works best at about 110°F. I found that it does not work well at all below 100°F.

Looking closely at the photos, you may notice that the IC sockets look a little odd. I designed the boards for SMT ICs. However, I did use mostly normal DIP devices. I found IC sockets with the flat side of their pins parallel to their long edges. By bending the pins to the side and extending traces at least 0.2 inches beyond the socket sides, the socket pins can be soldered on top of the traces. I had to trim the leads on a few sockets, but this is quite easy.

Mount ICs on both sides of the boards where necessary.

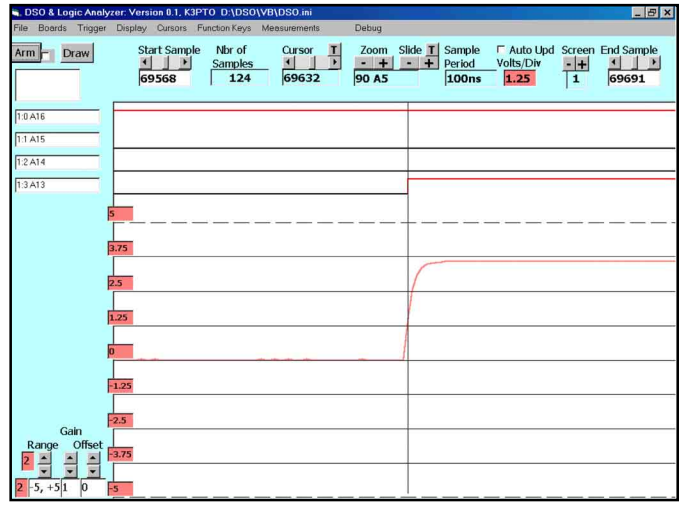

Fig 18—A view of A4 instead of A12. Also the time base has been changed from 1.6 µs per sample to 100 ns per sample. Notice that the signal fall-time can now be seen.

Some General Construction Information

Each board has several hand inserted feed-through wires and jumpers. All are labeled in copper. Each feed-through wire is labeled with an "F." The jumpers are labeled with the same lower-case letter at each end of the jumper. The silkscreen shows the locations of all resistors and capacitors. The leads of discrete resistors and capacitors often double as feed-through wires.

Be careful when soldering the board-to-board connectors. Do not get solder high up on the pin. If you do, it will be more difficult to connect to the board below.

The IC socket for the oscillator on the PLD board is the only one in the system mounted as a through-hole socket.

Cut each board the same size: 4.25×3.5 inches.

Insertion of the wires into the sockets for the right-angle connectors can be quite tedious, so I used wire removed from eight-conductor telephone cable. It is flexible and of eight different colors. At first, I soldered the wires after crimping them by hand, but sometimes the solder would wick up into the contact area. I did not solder the last several I wired, and they are holding up fine.

There are several connections from the Control board to the input boards. For several of these, I simply put heat-shrink tubing over the connector pin after crimping it rather than cut a housing for it.

Remember that you will loose a pin when you cut the board-to-board connectors to the appropriate size. File the cut side flat to make it look better.

Keep the main circuit functions on separate boards. This makes for a very modular system that can be easily customized. It also makes it easier to test the individual circuits. This was very important in the early designs, so that I could make changes. It is not quite as important now that most of the circuitry is in a PLD.

Install load resistors on selected inputs. This makes it easier to test the boards and protects CMOS devices, which can be damaged when their inputs float.

Install a few test points on each board—especially at least one ground. This makes debugging much easier! I even put a few test circuits into the PLDs of the first designs so I could monitor some internal signals.

Display Screens

Figs 16 through 18 are screen shots showing various features of the DSO/LA. Fig 16 shows four address bus sig-

Table 3—DSO/LA Parts, Listed by Board

Part	*Qty*	*Vendor*	*Part #*
Miscellaneous			
Cabinet, aluminum, 7.0×6.2×4.6 inches	1	Jameco	11893
Cable, 25 pin D, M-F, 5 feet	1	Jameco	163133
Connector, D Sub 25P, cbl, M	1	Mouser	156-1225
Connector, DIN, 5 pin F ch	1	Jameco	15843
Connector, DIN, 5 pin M cbl	1	Jameco	15878
Desoldering braid, 5 feet	1	Hosfelt	46-130
PW board developer	3	WebTronics	POSDEV
PW board: 12×12 inches, photosensitive	1	WebTronics	GD1212
PW board: etchant	1	WebTronics	
Rubber feet, 1/2-inch square by 1/8-inch high, 12 pcs	1	Hosfelt	85-109
Tubing, heat shrink, 3/32, 10 feet	1	Hosfelt	33-131
Wire, #30 wire wrap, 100 feet, red	1	Jameco	22630
Wire, miscellaneous			
Control			
Capacitor, 0.001 µF, 50 V	2	Hosfelt	95-179
Capacitor, 0.22 µF, 50 V	3	Hosfelt	82-102
Capacitor, 33 µF, 25 V	1	Hosfelt	15-455
Connector D Sub 25P, M, rt angle, PCB	1	Mouser	152-3325
Connector header, 36 pin, M, rt angle	1	Jameco	103270
Connector housing, 36 pin	1	Jameco	103157
Connector pin, female for 103157	20	Jameco	100765
IC Digital, 74HC138, 3-to-8 line decoder	3	Jameco	45330
IC Digital, 74HC245, 8 bit xcvr	1	Jameco	45671
IC Digital, 74HC574, 8 D FF, 3-state	1	Jameco	46084
Resistor network, 47 kΩ, 10P, SIP, common	2	Mouser	266-47K
Socket, 35 pin, single row	1	Samtec	ESQ-135-14-T-S
Socket, IC, DIP 16, order increments of 10	3	Jameco	112221
Socket, IC, DIP 20, order increments of 10	2	Jameco	112248
PLD			
Capacitor, 0.001 µF, 50 V	2	Hosfelt	95-179
Capacitor, 0.22 µF, 50 V	3	Hosfelt	82-102
Connector housing, 36 pin	1	Jameco	103157
Connector pin, female for 103157	4	Jameco	100765
Connector header, dual row, 10-pin male, right angle	1	Jameco	203932
Connector header, 36-pin, M, strip header	1	Jameco	103270
IC, PLD, 84 pin, PLCC	1	Arrow	EPM7128SLC84-15
Oscillator, 20 MHz	1	Hosfelt	23-132
Socket, 35 pin, single row	2	Samtec	ESQ-135-14-T-S
Socket, 84 pin, PLCC, through hole	1	Arrow	PLC C-84P-T-SMT
Socket, IC, DIP 14, order increments of 10	1	Jameco	112213
LA			
Capacitor, 0.001 µF, 50 V	2	Hosfelt	95-179
Capacitor, 0.22 µF, 50 V	2	Hosfelt	82-102
Connector D Sub 9P, F, cbl	1	Mouser	156-1309
Connector D Sub 9P, M, right angle, PCB mount	1	Mouser	152-3309
Connector D Sub 9P, hood	1	RadioShack	276-1539
Connector header, 36 pin, M, right angle	1	Jameco	68339
Connector header, 36 pin, M, strip header	1	Jameco	103270
Connector pin, female for 103157	12	Jameco	100765
IC Dig, 74HC00, NAND, Q2	1	Jameco	45161
IC Dig, 74HC541, 8-buffer, 3-state	2	Jameco	46050
IC Dig, SRAM, 128×8, SOJ, 25 ns	1	Avnet	CY7C109B-35VC
Resistor network, 1 kΩ, 16P, DIP, isolated	1	Mouser	652-4116R-1-1K
Resistor network, 47 kΩ, 10P, SIP, common	1	Mouser	266-47K
Resistor, 10 kΩ, 1/4 W, 5%	2		
Socket, 35 pin, single row	1	Samtec	ESQ-135-14-T-S

Part	*Qty*	*Vendor*	*Part #*
Socket, IC, DIP 14, order increments of 10	1	Jameco	112213
Socket, IC, DIP 20, order increments of 10	2	Jameco	112248
A/D			
Capacitor, 0.001 µF, 50 V	2	Hosfelt	95-179
Capacitor, 0.22 µF, 50 V	5	Hosfelt	82-102
Connector header, 36 pin, M, right angle	1	Jameco	103270
Connector housing, 36 pin	1	Jameco	103157
Connector pin, female for 103157	10	Jameco	100765
IC Digital, 74HC00, NAND, Q2	1	Jameco	45161
IC Digital, 74HC541, 8-buffer, 3-state	1	Jameco	46050
IC Digital, 74HC574, 8-D FF, 3-state	1	Jameco	46084
IC Digital, 74HC85, 4-bit comparator	1	Jameco	46172
IC Digital, SRAM, 128×8, SOJ, 25 ns	1	Avnet	CY7C109B-35VC
IC Linear, A/D Converter, 20-MHz, SOIC	1	DigiKey	ADS820U-ND
Socket, 35 pin, single row	1	Samtec	ESQ-135-14-T-S
Socket, IC, DIP 14, order increments of 10	1	Jameco	112213
Socket, IC, DIP 16, order increments of 10	1	Jameco	112221
Socket, IC, DIP 20, order increments of 10	2	Jameco	112248
Resistor, 10 kΩ, 1/4 W, 5%	3		
A/D Input Buffer			
Capacitor, 0.001 µF, 50 V	2	Hosfelt	95-179
Capacitor, 0.22 µF, 50 V	3	Hosfelt	82-102
Connector header, 36 pin, M, strip header	1	Jameco	103270
Connector housing, 36 pin	1	Jameco	103157
Connector pin, female for 103157	6	Jameco	100765
Connector, BNC, F, chassis	1	Hosfelt	952
Diode, 1N914, order increments of 10	2	Jameco	36311
IC Linear, op amp, 2× high speed, 1 Sup, SOIC	1	DigiKey	OPA2631U-ND
Pot, 500 Ω, 25 turn, 3/8-inch	1	Hosfelt	38-124
Resistor, 1 kΩ, 1/4 W, 5%	5		
Resistor, 1.7 kΩ, 1/4 W, 5%	1		
Resistor, 62 kΩ, 1/4 W, 5%	1		
Resistor, 47 kΩ, 1/4 W, 5%	1		

The quantities of some connector components will vary depending on the construction technique.

nals as captured by an LA board as well as the A12 signal as captured by an A/D board. Fig 17 shows the same signals as the previous screen shot but zoomed in. Fig 18 shows A4 instead of A12. Also the time base has been changed from 1.6 µs per sample to 100 ns per sample. Notice that the signal fall time can now be seen. Several more features appear in on screen, but some don't show in the B/W figures here:

- A different color for the A/D trace
- The analog voltage for the A/D channel is in the Status box near the upper left corner. The background color is the same as the one selected for the trace.
- There is another cursor on the screen. The system allows up to four more cursors in addition to the main one.
- Horizontal grid lines for the A/D signal along with the Volts/Div next to the Sample Period.
- The A/D Scale and Offset features

Larry has a BSEE from Drexel Institute of Technology and a Masters in Engineering Science from Penn State. He has worked at ZWorld in Davis, California, as a Technical Support Manager since March 2000. Before that, he worked for 30 years designing and developing automatic test equipment for automotive electronic subsystems.

Larry has been licensed since 1961, and currently holds an Advanced class license. He enjoys electronic hobbies, church and reading. □□