FEATURE ARTICLE

by Philip Nowe

Timing (Analysis) is Everything

A How-To Guide for Timing Analysis

Philip's main issue with young engineers is that many of them have been taught excellent circuit design techniques but haven't been schooled in the importance of timing analysis. What is timing analysis? Why is timing analysis important? How do you perform timing analysis? Whatever your level of expertise, you're sure to find Philip's answers informative.

As a hardware designer and manager, I've noticed that many electrical engineering students are often missing something when they begin their first full-time jobs. They've been taught how to design great circuits, some of them quite complex, but they haven't been taught the importance of timing.

What does timing analysis mean? Why is timing analysis important? How is it done? In this article, I answer these questions. In addition, I present you with a real design problem that was solved with timing analysis. So, here we go!

WHY TIMING ANALYSIS?

There are a couple of reasons for performing timing analysis. First and foremost, it can be used to verify that a circuit will meet all of its timing requirements. Timing analysis can also help with component selection. An example is when you are trying to determine what memory device speed you should use with a microprocessor. Using a memory device that is too slow may not work in the circuit (or would degrade performance by introducing wait states), and using one that is too fast will likely cost more than it needs to.

A WORKING DEFINITION

Timing analysis is the methodical analysis of a digital circuit to determine if the timing constraints imposed by components or interfaces are met. Typically, this means that you are trying to prove that all set-up, hold, and pulse-width times are being met.

A minimum or maximum digital simulation is not actually the worst-case analysis. That is what a number of entry-level engineers believe. The worst-case analysis takes into account minimum delays through some paths and maximum delays through other paths. For instance, the worst-case set-up timing with respect to flip-flop B in Figure 1 would be the minimum delay to the clock input combined with the maximum delay to the data input of flip-flop B.

Let's assume the timing values in Table 1 are for the circuit elements in Figure 1. Do you think that there is a problem with these values? Take a look at this circuit in a waveform view in Photo 1. Notice that the bottom of the photo shows the parameters used in determining the set-up and hold timing. Red indicates that a condition has not been met. If the set-up time is read and has a margin of –1, the set-up time has not been met and is off by 1 ns. The hold time indicates that there is 1-ns margin.

In Photo 1, the gray areas of the waveforms indicate the uncertainty of when the edge occurs. Notice that the output of logic gate 2 has the largest uncertainty, because the uncertainty is cumulative as you go through a delay chain. So, the delay at the output of logic gate 2 is equal to the delay from CLK A to Q of flip-flop A as well as the delays through logic gates 1 and 2. Note that the waveform also uses color highlighting to indicate that constraints are not being met.

As you can see in Photo 1, there is a D input set-up time problem to flip-flop B. Sometimes, when discussing timing issues, I hear designers say that timing doesn't really matter because the processor has a memory controller with variable timing. This may be true, but it usually means that the processor allows for a programmable number of wait states. If you add another wait state (i.e., one more clock cycle before clocking in the data), then the problem in Photo 1 will go away. But what if you don't want the performance hit of adding a wait state, or what if the processor doesn't allow wait states? You

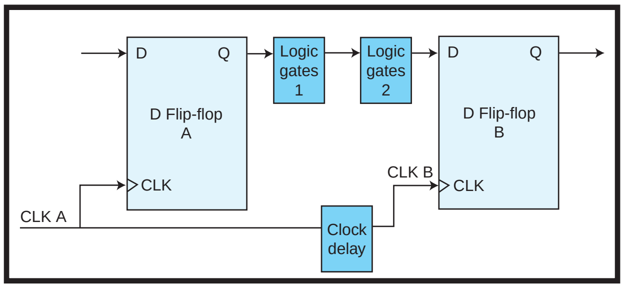


Figure 1—*The simplified digital circuit contains delays in the data and the clock paths. The timing values are shown in Table 1 (see p. 29).*

would have to solve the timing problem.

Another case involves hold problems. Adding wait states often cannot solve this, because the timing chain for the D input is tied to the current clock edge not to the delays from the previous clock edge. In such a case, you need to make some changes to the design to make the timing work.

Okay, so you agree that there is a problem. So what? What will happen if you don't fix it? There are three possibilities for set-up and hold times (see Figure 2).

As you can see in Figure 2a, the signal of interest can meet the timing with proper set-up and hold times. The next possibility is that the signal may miss it completely and get caught on the next clock edge (see Figure 2b). (Note that this can be a problem if you don't want the performance penalty.)

The last possibility is that the input signal changes inside of the set-up and hold window (see Figure 2c). What happens in this case? The output of the flip-flop can become metastable, which means that the output can oscillate from zero to one or from one to zero a few times (or many times) before it stabilizes to a zero or one. The resulting state is random. (For more information on metastability, refer to H. Johnson and M. Graham's book, *High-Speed Digital Design: A Handbook of Black Magic*). Obviously, this is not a good situation, because the output of the flip-flop may be wrong, and it may take longer than the normal propagation delay to get to the wrong value.

Knowing that you have a problem is the first step. So, how can you fix it? There are many ways to solve timing problems. In this simplified circuit, you are off by 1 ns. You can change either logic gates 1 or 2 so that they are faster parts. Another option is to select a flip-flop that has a smaller set-up and hold window. Timing analysis doesn't fix the problem; it just tells you that there is a problem. Remember, when you make a change to your circuit, rerun the timing analysis to make sure that the problem is fixed and that another one hasn't been created. Hopefully, I have convinced you that timing analysis is important. Now I'll show you how to do it.

Timing parameter	Minimum value	Maximum value
Nominal CLK frequency	25 MHz	25 MHz
CLK to Q delay (both flip-flops)	2 ns	5 ns
Clock delay	1 ns	3 ns
Propagation delay through logic gates 1	3 ns	15 ns
Propagation delay through logic gates 2	5 ns	12 ns
D input setup time to CLK (both flip-flops)	10 ns	
D input hold time after CLK (both flip-flops)	6 ns	

Table 1—*Here are the timing values for the circuit illustrated in Figure 1.*

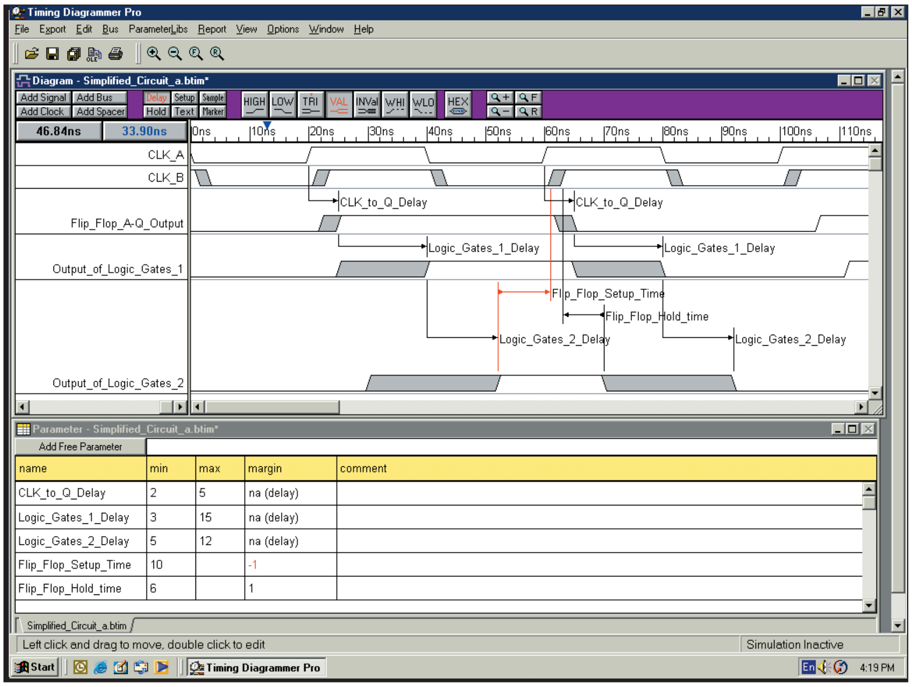
HOW IS IT DONE?

Timing analysis has been achieved in many different ways over the years. You can use anything from a manual approach (i.e., using spreadsheets and a drawing program or just pen and paper) to what I refer to as semimanual CAD programs. You can also use fully automatic static and dynamic timing-analysis tools.

I am visually oriented like most engineers, so I prefer to draw my timing diagrams. For the first board I developed, I used my schematic drawing tool to draw the timing waveforms. For the signal timing, I put on the diagram the minimum and maximum timing for every signal edge. Each time I changed the components in a signal path, I updated the numbers on the drawing. The next possible step in the evolution of timing analysis would be to put the timing numbers in a spreadsheet and let the spreadsheet do the calculations.

Table 2 shows the original simplified circuit analyzed in a spreadsheet format. For the set-up time calculation, use the maximum data delay and the minimum clock delay (less set-up time) to determine if the set-up time is met. For the hold time, use the minimum data hold delay and the maximum clock delay plus the hold time to see if the hold time is met. This is straightforward but time-consuming. What you need to do is to calculate each of the signal paths going to flip-flop B, for instance. The advantage of using a spreadsheet is that it saves you time when making changes to the design. A combination of a spreadsheet and a drawing seems like the right way to go.

I'm familiar with two popular semimanual timing analysis products:



name	min	max	margin	comment
CLK_to_Q_Delay	2	5	na (delay)	
Logic_Gates_1_Delay	3	15	na (delay)	
Logic_Gates_2_Delay	5	12	na (delay)	
Flip_Flop_Setup_Time	10		-1	
Flip_Flop_Hold_time	6		1	

Photo 1—*I used Timing Diagrammer Pro for the timing analysis of the simplified digital circuit. Note that the gray areas on the waveform denote regions of uncertainty. The red areas show a timing violation.*

SynaptiCAD's Timing Diagrammer Pro and Forte Design's Systems TimingDesigner. These two products are roughly similar. The timing diagrams in this article use Timing Diagrammer Pro.

The Timing Diagrammer Pro is a timing analysis tool designed to assist the digital designer in modeling and analyzing digital circuits. (Another tool from SynaptiCAD is Waveformer Pro, which also allows you to export waveforms as VHDL or Verilog for simulation purposes.) It has two main windows for analysis, the first of which is the diagram editor window where you draw the waveforms. There are special tools to help with clocks, and you can create waveforms from other waveforms. You can do so with a Boolean equation—(SIG0 and SIG1 and SIG3) delay 20 ns—or it can be specified using VHDL or Verilog. The other important window is the parameter window, which is like the aforementioned spreadsheet; it holds the timing parameters of the design. The power of Timing Diagrammer Pro is that the parameter values and the timing waveforms are linked.

TIMING DIAGRAMMER PRO

Performing a timing analysis using Timing Diagrammer Pro is straightforward. First, you need to draw in the waveforms. Initially, there are no delays or constraints; you don't have to be accurate at first, because the accuracy comes when you add the delays and constraints.

Second, you need to add the delay and constraint information to the waveforms; this will automatically add the delays and constraints to the parameter spreadsheet. Then, enter the exact minimum and maximum numbers for the delays and constraints in the parameter spreadsheet. Doing so automatically updates the view in the diagram editor window and shows the areas of timing uncertainty.

Timing Diagrammer Pro allows you to use libraries of timing values for parts. Thus, you can reuse some of the work that you've (or someone else) done already.

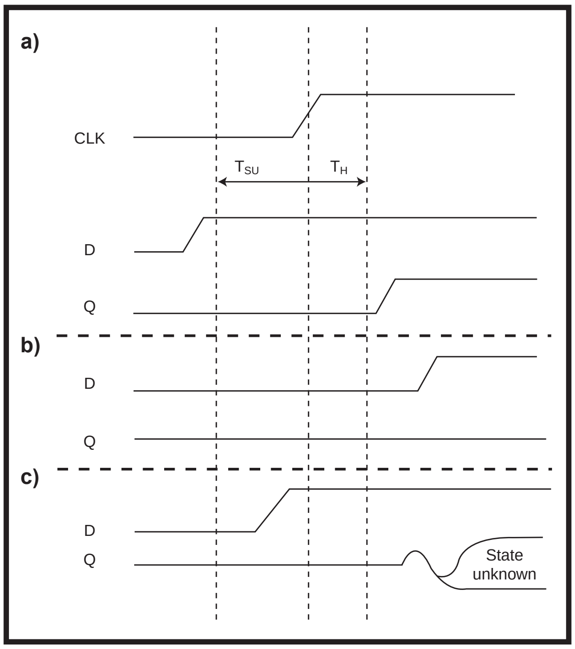


Figure 2a—*Data arrives before the set-up time requirement. Data is clocked into the flip-flop on the rising edge of CLK.* **b—***Data arrives after the hold time, which results in the data being clocked into the flip-flop on the next rising edge of CLK.* **c—***Data arrives within the set-up and hold window, which results in an indeterminate output from the flip-flop.*

DESIGN SEQUENCE

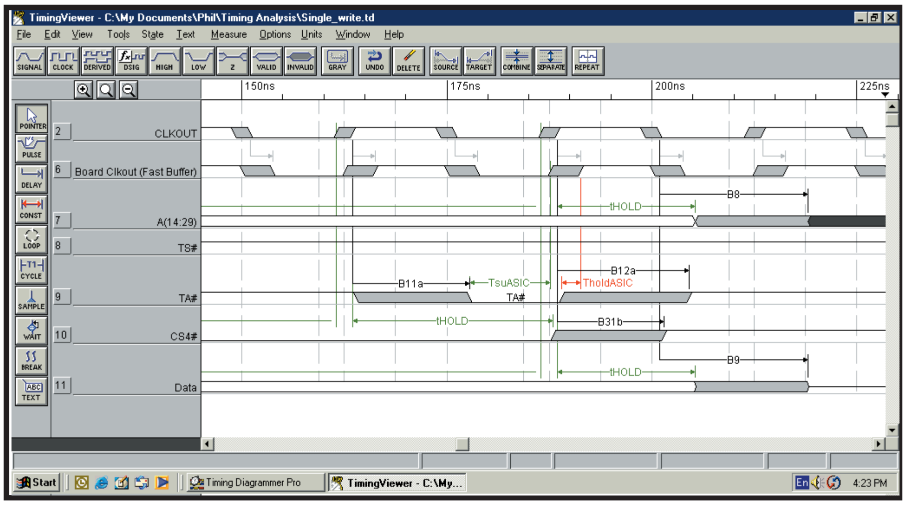
You now know how to use Timing Diagrammer Pro to perform timing analysis, but how can you use the tool in a real project? The following sequence has worked well for my team for a number of design projects.

First, capture the interface specifications in Timing Diagrammer Pro with all of the timing constraints shown. An interface is any part of the design that interacts with another part, such as a write cycle from a microprocessor to a memory and a connection to a PCI bus. These interface specifications form the basis for subsequent design decisions; they may give the designers an early indication as to whether the design is feasible, impossible, or sheer lunacy. For instance, if the interface specifications dictate that you will have to use a 34-ps SRAM, you'll probably try to get on another design project!

As the design progresses, put real timing numbers into Timing Diagrammer Pro, which will immediately tell you if the constraints are still met. At

Set-up time calculation						
Data delay				Clock delay		
Timing parameter	Minimum (ns)	Maximum (ns)		Timing parameter	Minimum (ns)	Maximum (ns)
CLK A or CLK B to Q delay	2	5		CLK A or B period	40	40
Propagation Delay1 through logic gates	3	15		CLK A to B delay	1	3
Propagation Delay2 through logic gates	5	12				
			Minus	D input setup time to CLK1	10	10
Total data propagation delay	10	32		Time from CLK to CLK1 accounting for set-up time	31	33
				Slack for set-up time	–1	
Hold time calculation						
Data hold				**Clock delay**		
Timing parameter	**Minimum (ns)**	**Maximum (ns)**		**Timing parameter**	**Minimum (ns)**	**Maximum (ns)**
CLK A or CLK B to Q delay	2	5		CLK A to B delay	1	3
Propagation Delay1 through logic gates	3	15		D input hold time after CLK B	6	6
Propagation Delay2 through logic gates	5	12				
Total data hold time beyond CLK B	10	32		Time from CLK to CLK1 accounting for set-up time	7	9
				Slack for hold time	1	

Table 2—*For the simplified circuit, the set-up time slack is equal to the minimum clock delay minus the maximum data delay. The hold time slack is equal to the minimum data delay minus the maximum clock delay.*



TimingViewer - C:\My Documents\Phil\Timing Analysis\Single_write.td
150ns
175ns
200ns
225ns
CLKOUT
Board Clkout (Fast Buffer)
A(14:29)
TS#
TA#
CS4#
Data
tHOLD
B8
B12a
B11a
TsuASIC
TholdASIC
B31b
B9

Photo 2—*Note the width of the Board Clkout signal. This is the result of it being an ordinary buffer that is heavily loaded on the board. Again, red indicates a timing violation.*

some point, there may be more than one option. Using this tool, you can model each of the possible solutions to determine if they work. If they do not work, then redesign, whether that means simply changing to a faster component or a completely new circuit. An alternative might be to change the original requirements, assuming that your customer allows you to do so. Don't count on it!

When it comes time to review your design prior to building your PCB, for instance, if the Timing Diagrammer Pro files are up to date, you have proof that your design will meet the timing requirements. Note that if your circuit has critical timing paths, you may want to include PCB delays in the timing analysis as well.

AUTOMATIC CAD TOOLS

ASIC designers have been using static timing analysis tools for a long time. Synopsys's Primetime is an example. The tools go through the entire design and determine if there are any timing violations (with some constraints from the user to minimize false paths). There are static timing analysis tools for board-level design, as well (e.g., BLAST, which was developed by Innoveda). These tools generally cost significantly more than Timing Diagrammer Pro and Timing Designer.

What if a tool shows that you are in error? Is this always true? You may say that timing analysis is too pessimistic and, at times, you may be right. For instance, if the elements in the simplified circuit depicted in Figure 1 are in one FPGA, it is less likely that the data path will exhibit a maximum delay and the clock path will exhibit a minimum delay. This is because they are in the same part, and delays on a chip tend to track each other. The data path may be at the maximum delay but the clock delay will be too. For most other cases, however, it is best to use worst-case timing numbers.

A PROBLEM SOLVED

At a former employer of mine, we had a problem with a card we were working on. We couldn't write to some of the address space in one of our ASICs on a new spin of the card. So, after playing with the software to make sure that it was not the cause of the problem, we hooked up the logic analyzer to see what was going on. At first glance, the timing looked fine, and we scratched our heads. But just before we went home late that evening, one of the ASIC designers said that he found it funny that the first write cycle that had worked was the only one that worked. It would have been nicer if he had mentioned that earlier!

When we returned to the lab the next day, we concentrated on looking at the second and subsequent write cycles. One of the control signals to the ASIC was rising at the same time as the clock. Our hypothesis was that the timing wasn't OK.

We decided to try moving the clock signal in time by delaying the clock. We initially did this by adding a long wire to the clock signal. As a result, the write cycle worked! Well, it mostly worked. Next, we used a footprint-compatible buffer, which was slower and seemed to work well enough for the card to be used by the SW developers and other testers to continue with their work. However, it didn't explain exactly what was wrong. Why was the control signal so close to the clock signal? Why hadn't we seen this on the previous version of the card?

We went back to the timing analysis for the previous version of the card, and it showed that there shouldn't have been any problems. On closer inspection, however, we noticed that the analysis was done to the wrong clock edge. When the timing analysis was changed to the correct clock edge, it immediately

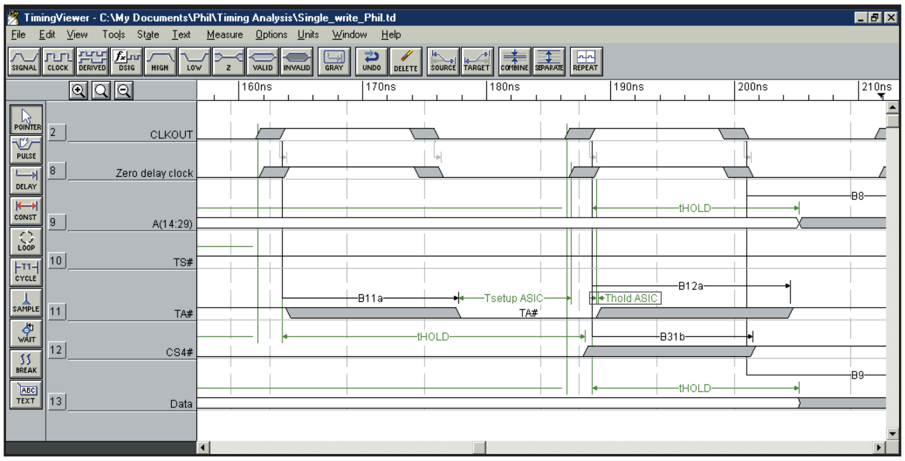


Photo 3—*I used a zero-delay clock driver. The area of uncertainty on this clock is significantly less than the Board Clkout signal in Photo 2. Notice the lack of red this time. It works!*

flagged that there was a problem.

Why wasn't there a problem with the previous version of the card? After talking to the software folks, we found out that the same write cycle on the previous version of the card didn't work either! They had found a way around it, so they didn't complain too loudly. At that point, we knew that we had a problem that needed to be solved on both circuit cards. The problem with the timing in the circuit was that the clock was being delayed quite a bit because of the load on the clock. The new version of the card added two more loads to the clock line, which, in turn, caused the clock to rise more slowly and arrive coincidently with the control signal.

We looked at the buffer and found another part that was a zero-delay buffer, which meant it had a PLL in it to synchronize the output clocks with the input clock. We put the timing numbers from the new part in the timing analysis, and it worked. We then dead-bugged the part on the board, which was not an easy feat with a number of BGAs. After we had solved a few other problems with the ASIC, the board worked.

Photo 2 shows the timing analysis for the circuit with the old clock buffer. Photo 3 depicts the timing analysis with the zero-delay buffer. (I used Forte Systems's Timing Viewer, because the analysis was performed in Timing Designer.)

CONVINCED YET?

I hope I've convinced you of the importance of timing analysis, which you can now perform manually or with a semiautomatic CAD tool. Remember, whether you have a design running at 1 MHz or 1.5 GHz, timing matters! ◼

Philip Nowe earned a Bachelor's in Engineering at Carleton University in Ottawa, Canada. He has been working in the hardware design industry for the past 20 years. He has experience in board design, PLD/FPGA design, and hardware management. Currently, Philip is a digital design consultant. You may contact him at pnowe@sympatico.ca.

RESOURCE

H. Johnson and M. Graham, *High-Speed Digital Design: A Handbook of Black Magic*, Prentice Hall, Upper Saddle River, NJ, 1993.

SOURCES

Timing Designer
Forte Design Systems
(800) 585-4120
www.forteds.com

BLAST
Mentor Graphics Corp. (Innoveda)
(800) 547-3000
www.innoveda.com

PrimeTime
Synopsys, Inc.
(800) 541-7737
www.synopsis.com

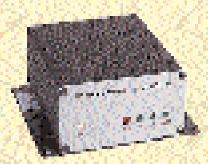
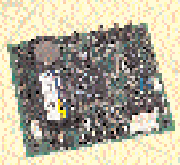
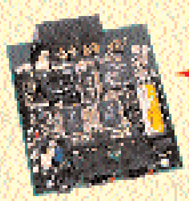
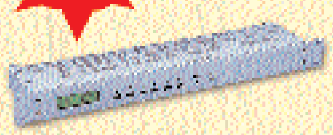
Timing Diagrammer Pro and Waveformer Pro
SynaptiCAD, Inc.
(800) 804-7073
www.synapticad.com

Tools to Move Data

RS-232 <> Ethernet <> Web Server

TCP/IP Solutions

NEW Products

µFlashTCP-EP*	LogicFlex*	Dual-E*	LogicFlex-EPX*
10Base T Ethernet	10 Base T Ethernet	2 Ethernet Ports	10Base T Ethernet
2 Serial Ports	2 Serial Ports	2 Serial Ports	2 Serial Ports
7-34 VDC	46 Digital I/O's	3 Counter/Timers	16 Opto Isolated inputs
Optional I/O Interface	Programmable Xilinx CPLD	5 Digital I/O's	16 Relay Outputs @ 500mA ea.
Rugged Enclosure w/ Industry Standard Connectors	Hardware Clock/Calendar	Onboard Connectors	Quick Disconnect Connectors
	Expansion Bus for Peripheral Boards	USNET TCP/IP Software in Kits	Rack Mount Enclosure
			LCD & 6 Pushbuttons
Starts at $229	Starts at $189	Starts at $199	Starts at $499

*All products based on the Intel 386Ex with DOS and NE2000 compliant, 10Base-T Ethernet. Standard memory includes 512K RAM & 512K Flash plus DIP socket to accept an M-Systems DiskOnChip. Development systems contain necessary hardware and software tools for fast development.

JK microsystems connects you with embedded control solutions. Our cost-effective DOS based controllers are ideal for data acquisition, networking or industrial technology applications.

JK microsystems, Inc. 1403 Fifth St, Suite D Davis, CA 95616 USA Visit us on the web www.jkmicro.com Call 530-297-6073 Fax 530-297-6074