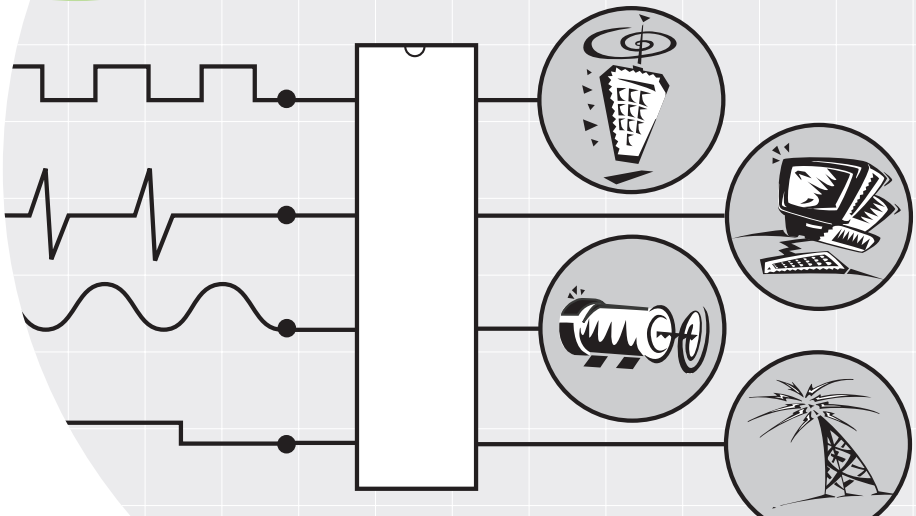
8 Hints for Debugging Microcontroller-based Designs

Time-saving tips from successful designers of 8- and 16-bit systems.



Agilent Technologies
Innovating the HP Way

Introduction

Contents

The challenge of debugging MCU-based designs

It's almost impossible to design an electronic or electromechanical product these days without using a microcontroller. While there are plenty of interesting design challenges, the debugging tools for MCU-based designs haven't always kept up.

If you work with 8- and 16-bit MCUs, for instance, you've probably felt stuck in the middle, between generic, basic tools (such as scopes) and higher-end tools aimed at microprocessors (such as traditional logic analyzers and emulators). At the same time, you're probably dealing with a mix of analog and digital signals, so a scope by itself or a logic analyzer by itself is only half a solution.

Moreover, you probably don't have the luxury of specializing. You have to know analog hardware, digital hardware and firmware—and be good at all three. And all the while, market windows are getting narrower, competition is getting stronger, and customers are expecting more power and capability from your MCU-based products. Your job may be a lot of things, but boring certainly isn't one of them.

Help is on the way

As the worldwide leader in test and measurement, we're working hard to help engineers like you meet your MCU-based design challenges. One way we can help is with the information in this booklet, practical debugging hints from engineers working with a variety of MCUs. You'll see how these designers use some of the latest MCU debugging tools to get their new products to market faster.

Get FREE Agilent test equipment

Would you like to share your MCU debugging experience with thousands of engineers worldwide? Submit a hint like the ones you'll find in this booklet, and if we use it in a future application note, we'll say thanks by providing you with free Agilent test equipment valued up to US $1000. If you're interested, please email us at **dear-scopie@agilent.com**.

Tracking down elusive glitches

Hint 1

By Steven Schram, Invocon. Inc.

Infrequent, unpredictable events can present some of the toughest troubleshooting challenges around. I recently encountered such a glitch while designing a low-power data acquisition device. This wireless instrument system uses a group of remote sensor units and spread spectrum radio transceivers (Figure 1). Data collected at the system can be retrieved by a network control unit connected to a computer.

The system uses the interrupt pin on a low-power clock chip to trigger power-up events every 60 seconds. Between events, the clock and supporting logic are the only devices drawing current (approximately 50 µA). After getting a trigger from the clock, the Motorola 68HC11K1 microcontroller powers up, collects temperature data and listens for transceiver activity. If it hears a data request on the transceiver, the MCU transmits the temperature data.

The glitch in question was showing up during this 60-second interval, when the system was supposed to be quiet. To find and analyze this anomaly, I used a deep memory (1 Mbyte) digital oscilloscope with peak detect. Since the glitch occurred so infrequently, I first set the scope's time base at 10 seconds/division in order to capture the entire 60-second sequence. Without peak detect, most narrow events would be impossible to detect at this time base setting. But as Figure 2 shows, the glitch in my system was easily captured and viewed. Peak detect showed something unusual happening approximately 15 seconds after the clock trigger event.

Once I became aware of the anomaly's presence in my system, the scope's deep memory made it easy to zoom in and analyze the glitch in more detail. With the scope's 1 Mbyte of acquisition memory, the initial waveform capture at 10 seconds/division was also sufficient to see waveform details when I zoomed in and viewed at 10 milliseconds/division (Figure 3).

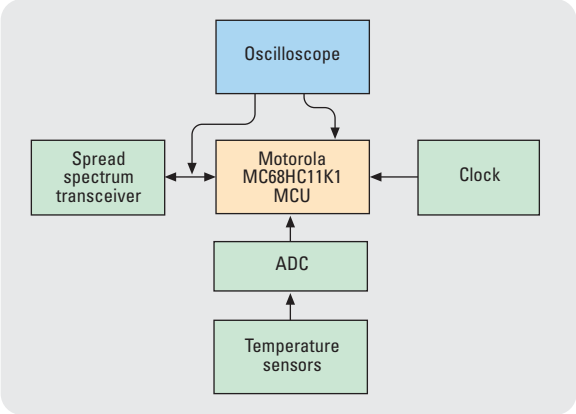


Figure 1. *Basic block diagram of the wireless instrumentation system. One analog input on the scope monitored power-up on the MCU while the other monitored the carrier detect signal feeding the transceiver.*

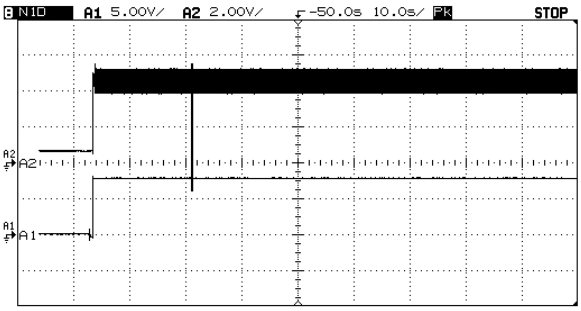


Figure 2. *The initial measurement using peak detect with a 10 s/div time base setting.*

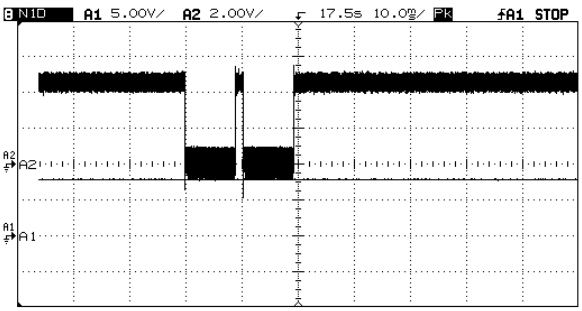


Figure 3. *Once the glitch was identified, zooming in to a faster time base setting provided the necessary details.*

Hint 2

Characterizing analog-to-digital converters using a PC-hosted logic analyzer

by Steve Warntjes, Agilent Technologies

With today's digital revolution in the electronics industry, analog-to-digital converters (ADCs) are increasingly common. ADCs can be found as either stand-alone parts or integrated into microcontrollers as special I/O peripherals. Depending upon the analog input accuracy requirements of your particular MCU-based application, certain ADC specifications may require verification prior to releasing the design to production. This hint shows how to test analog-to-digital conversion integrity using a low-cost oscilloscope, an arbitrary waveform generator, and a PC-hosted logic analyzer (test setup shown in Figure 1). Specifically, this hint will examine differential nonlinearity errors as they relate to missing digital codes of the ADC.

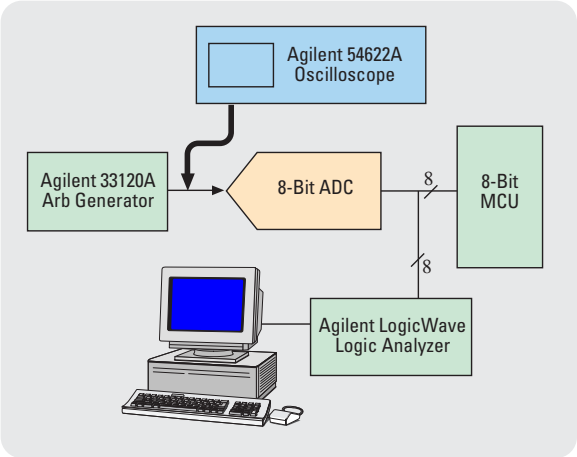


Figure 1. *ADC test setup.*

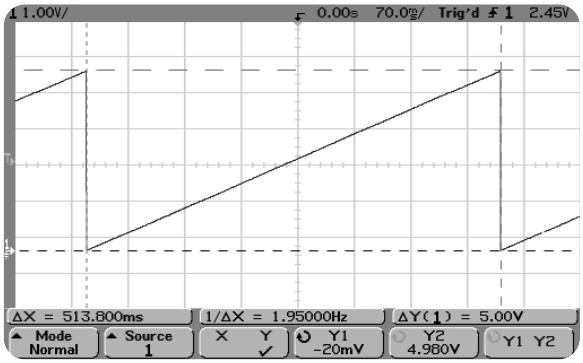


Figure 2. *The oscilloscope measured ADC test signal generated with an Agilent 33120A.*

Generating the test signal

A common test signal used to evaluate analog-to-digital converters is a voltage ramp. The voltage range for the ramp is the range of analog inputs applied to the converter for which it is expected to generate digital outputs. In this particular example where we test the integrity of an 8-bit ADC, the operating range is 0 to 5 volts. Because an ADC converts continuous analog voltages to a discrete number of digital codes, a conversion (or quantization) error will be introduced. In our case, each of the 256 possible digital output values represents a 19.5-millivolt voltage range for the analog input.

In this example, we are testing for missing digital output codes. To ensure that the ADC has an opportunity to output the correct code for the given analog input value, we generate the voltage ramp slowly enough that the ADC has at least four chances to output each digital code. For this example, the input voltage change is about 19.5 mV every four sample clocks. Since our sample clock is 2 kHz, four sample periods represent 2 milliseconds. Applying this voltage change over the entire 5-volt operating range results in a 1.95 Hz ramp.

We also have to consider test signal noise on the ADC input. An input signal with 50 millivolts of noise could generate missing ADC output codes even if the ADC is operating correctly. Figure 2 shows an oscilloscope measurement using an Agilent 54622A digitizing scope. The test signal was generated from an Agilent 33120A function/arbitrary waveform generator. A close examination of an expanded portion of the signal shows that we have about 10 millivolts of noise. If the noise is significant enough to cause missing ADC codes, multiple applications of the ramp can be applied. If the noise is truly random, eventually we would expect to see all apparent missing codes.

Capturing the ADC digital outputs

Most logic analyzers have two modes of operation: synchronous/state analysis and asynchronous/timing analysis. The asynchronous sampling mode will produce timing waveforms, much like an oscilloscope. The synchronous sampling mode produces a state listing of captured data on most logic analyzers. To test output codes of ADCs, the synchronous/state mode is the preferred sampling method to use. In this mode of operation, the analyzer samples digital bus values synchronous to a clock edge that is provided by the

system under test. In our example, the analyzer's assigned sample clock is the ADC sample clock. The advantage of this method of acquiring data (as opposed to asynchronous/timing analysis mode) is that the data is sampled only when data is stable and valid. In synchronous/state analysis, the data must be stable for the logic analyzer to sample correctly. This stability time is referred to as the setup/hold window of the logic analyzer. In our example, the rising edge of the sample clock allows us to capture the data from the previous analog input. The ADC data sheet usually specifies a data setup and hold specification relative to the sample clock on the data outputs. To ensure measurement integrity, the ADC setup/hold specifications should exceed the logic analyzer setup/hold requirements.

Figure 3 shows a logic analyzer display (state listing) of the ADC output. The primary or top window shows the logic analyzer triggering on the first 00H ADC output. Note that the ADC output label counts up from the 00H value. Also, note the absolute time displayed in the far right column. The time between the X and O markers shows the four samples that were collected while the ADC output value was 2.

To look for missing ADC codes, a logic analyzer's trigger capability is invaluable. Set the logic analyzer to trigger on a specific ADC code, such as 43H. If the logic analyzer never triggers, we can be sure that the ADC output is never hit. By applying a continuous set of input ramps with a specific logic analyzer trigger condition, we can quickly determine if there are any missing codes even if the noise on the ADC input is larger than the ADC step as discussed above.

Analyzing the captured data

After the data has been captured, it is a simple matter to process the information with a tool such as Microsoft® Excel. From within the Agilent LogicWave logic analyzer (Figure 3) we can save the captured data and easily import it into Excel. Once in Excel, the captured data can be charted versus an ideal voltage ramp. Figure 4 shows a portion of the voltage ramp comparing the ideal voltage to the ADC output. Note that a specific ADC code is missing, resulting in differential nonlinearity error. With the use of PC software, the oscilloscope waveform can also be captured and displayed. Comparing multiple views of the ramp, the ideal ramp, the oscilloscope measured ramp and ADC output measured with the logic analyzer, is a good way to analyze ADC conversion.

Microsoft is a U.S.-registered trademark of Microsoft Corporation.

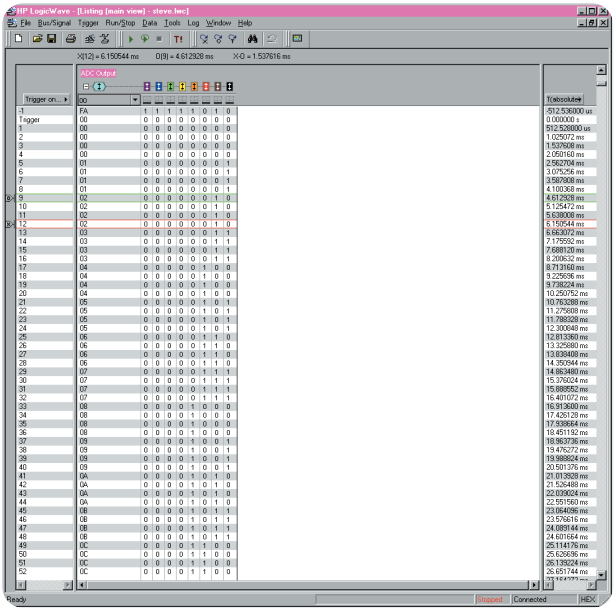


Figure 3 *The Agilent LogicWave synchronous capture display of the ADC output codes.*

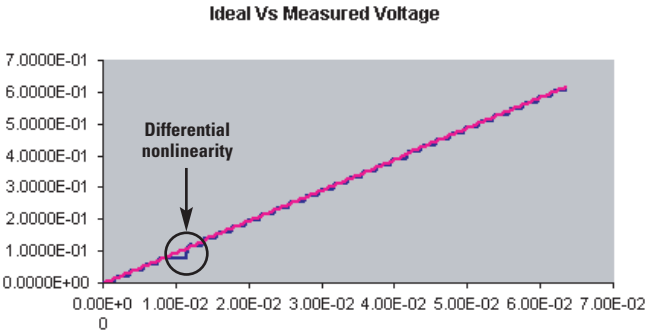


Figure 4 *A comparison of the ideal ADC voltage ramp with the Excel charted ADC output.*

3 Hint

Verifying PWM dead time in motor controllers

Technical staff, Infineon Technologies

Generating pulse width modulated (PWM) signals with an MCU is a common way to control AC motors with sine-wave shaped currents. A typical application for an 8-bit MCU is controlling a three-phase induction drive with variable speed in an open-loop configuration.

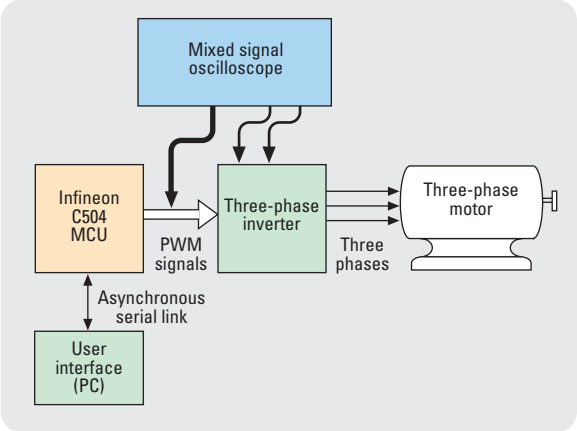


Figure 1. *Block diagram of an open-loop configuration for generating safe PWM signals to drive a three-phase motor.*

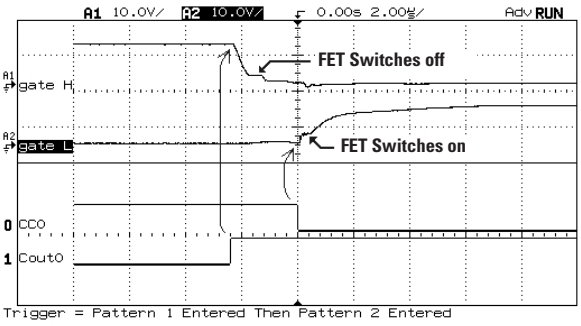


Figure 2. *Verifying the dead time between the high-side and low-side PWM outputs.*

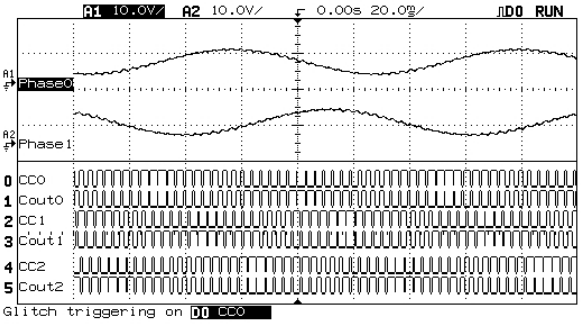


Figure 3. *Monitoring all six PWM signals and the phase currents on the high-side and low-side switches.*

However, the MCU can't drive an induction motor directly, so you need to amplify the three-phase signals first. Instead of using analog amplifiers, a more efficient way is to digitally amplify the PWM outputs with power switches, such as MOSFETs or IGBTs. The three-phase inverter shown in Figure 1 accomplishes this function.

The hardware for each phase of the inverter consists of two power switches (high side and low side) in a push-pull configuration. This creates a potential problem, though, if the control signals for the switches are exact complements of each other. During PWM switching, both power switches might momentarily conduct simultaneously due to different transistor turn-on and turn-off latencies. This can create a high-current short circuit and may destroy the inverter. It's therefore important to use an MCU optimized for motor control, such as the Infineon C504 (an 8051 derivative) or C164 (16-bit architecture). Both can be programmed to insert "dead time" in the PWM outputs by hardware without any software overhead. The dead time ensures that the two switches never conduct at the same time.

After programming the microcontroller to create the PWM output signals with dead time, the next step is testing the wave shape and timing. A four-channel scope can do the basic measurement, but if one is available, a mixed signal scope is a better choice because you can measure multiple analog and digital waveforms simultaneously and set up complex logic triggers.

Figure 2 verifies that the programmed dead time is sufficient for safe PWM switching. This zoomed-in display shows the impact of the dead time on the analog gate-source voltage of the power switch MOSFETs. The scope's cursors simplify the correct timing measurement and help characterize the circuit precisely.

With combined digital and analog measurement channels, you can easily monitor all six PWM signals and the phase currents. Figure 3 shows the two phase currents and corresponding digital PWM pattern. The time-qualified trigger mode lets you synchronize the scope's display to an adjustable pulse width corresponding to a well-defined phase angle.

Verifying I²C bus arbitration

by David Brobst, Solutions Cubed

The I²C™ bus allows multiple microcontrollers to share resources over a single communication channel. I²C is a synchronous, bi-directional, multi-device communication bus. The real strength of the I²C bus is that it only requires two wires for communication: data (SDA) and clock (SCL). Multiple devices can be attached to these two lines, thereby easing connection issues. However, this advantage adds a layer of complexity to communication exchanges on the bus. Problems can occur when two devices try to communicate at the same time.

The I²C protocol is set up in a master/slave configuration. A master must initiate all communication and control the clock signal. Once a master starts communication, all other masters refrain from starting communication. However, problems can arise when two or more masters try to start communication at the same time. Thankfully, I²C has a method of self-arbitration built into the protocol.

Both the SCL and the SDA lines are configured in an open-collector wired-AND manner. This means a device that outputs a '0' will override any other device trying to output a '1'. Masters on the bus monitor the actual state of the SCL and SDA lines and compare the bits on those lines to the ones they are trying to output. If at any time there is a mismatch, the master knows another master is on the bus, and it stops trying to communicate.

Many engineers working with embedded systems have wrestled with adding arbitration into their I²C firmware designs. Covering all cases can become a headache, and quickly add to the size and complexity of the firmware. Microchip has eased this burden with their new PIC18CXXX line of microcontrollers, which include full master support for the I²C bus, implemented in a hardware synchronous serial port (SSP). Using the SSP to control I²C communication relieves you of the most onerous aspects of I²C arbitration.

Figure 1 shows a typical system where master mode arbitration comes into play. In this case, both masters share the I²C bus and can try to communicate to either the EEPROM or the temperature sensor at any time. If there was no arbitration, data could be lost. However, the PIC18C452 controller provides a flag indicating that a bus collision has occurred. Writing firmware to utilize this bit is a relatively simple matter. If the bit is ever set, a different master has control of the bus and the controller has stopped all I²C action. The system can try again when the I²C bus is again free. Figure 2 shows a code fragment that accomplishes this task.

When communicating to the EEPROM over the I²C bus, the first byte sent is a $A0_H$. Likewise, the first byte sent to the temperature sensor is a 90_H. If one master starts communication to the EEPROM and the other to the temperature sensor at the same time, the controller talking to the EEPROM should lose arbitration on the third bit of the first byte. It loses arbitration because of the wired-AND configuration of the bus (the third bit to the EEPROM is a '1' and to the temperature sensor is a '0').

Using Agilent's new 54622D mixed signal oscilloscope (MSO), you can easily verify the arbitration process in the system shown in Figure 2. By using this oscilloscope's I²C triggering capabilities to trigger on a start condition, you can capture the beginning of an I²C communication. Figure 3 shows the capture of Master #1 starting communication with the temperature sensor, while Master #2 is starting to communicate to the EEPROM. Master #2 toggles an I/O pin twice any time its BCLIF bit gets set. The digital channel D1 on the Agilent 54622D MSO shows this I/O. As shown in Figure 3, Master #2 indicates its BCLIF is set during the third bit of the I²C communication, signaling it has lost arbitration and is no longer on the I²C bus. At this point, Master #2 must wait until Master #1 is done with communication before trying to access the I²C bus again.

Because it only uses two I/O lines, the I²C bus is a valuable tool to use in your embedded designs. With Microchip's new PIC18CXXX line of microcontrollers, it is no longer a daunting task to create the firmware you need for implementing I²C routines.

I²C is a registered trademark of Philips Corporation

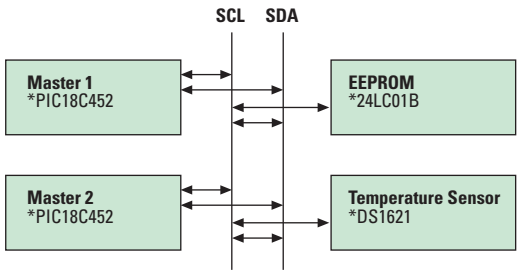


Figure 1. *I²C System*

```
Collision_Monitor
      btfss    PIR2,BCLIF     ;If collision, wait till over
      goto     Collision_Monitor_end
Collision_Wait
      btfss    SSPSTAT,P      ;Bus is idle when a stop bit is detected
      goto     Collision_Wait
      bcf      SSPSTAT,P      ;No spurious future stop bit detection
      bcf      PIR1,SSPIF     ;Stop condition sets SSPIF so clear
Collision_Monitor_end

      <Remaining I²C code goes here>
```

Figure 2. *I²C arbitration in the PIC18CXXX*

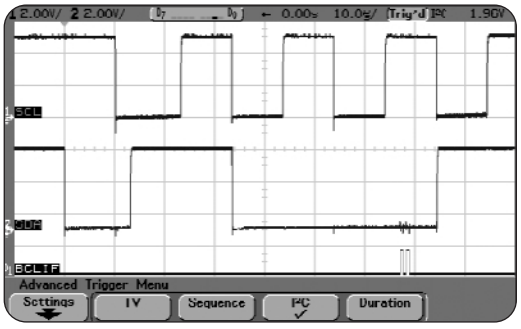


Figure 3. *Arbitration loss by master 2.*

5 Hint

What-if testing for MCU debugging

Jim Clark, LPA Designs

One of our biggest challenges as designers is verifying that our devices will work as well in the messy real world as they do in the lab. This is particularly important when a design needs to handle unpredictable analog signals.

Our digital remote controllers (Figure 1), which are essentially digital radios, live in an environment of noisy, often corrupted, signals. As a result, the data reception software built into the receivers needs to handle a variety of signal conditions appearing at the received strength signal indicator (RSSI). When we test over short distances on a lab bench, however, reception is usually too good to encounter random bit errors. And when an error does occur, it is usually not very repeatable.

We found a simple solution in an arbitrary waveform generator, which can reproduce virtually any waveform that we can represent as a set of time/voltage pairs. The first step is digitizing a clean and verified waveform that represents a good digital data packet. Any digital scope can perform this step, although we use a deep-memory, mixed-signal oscilloscope (MSO) since we need to capture a combination of digital and analog signals.

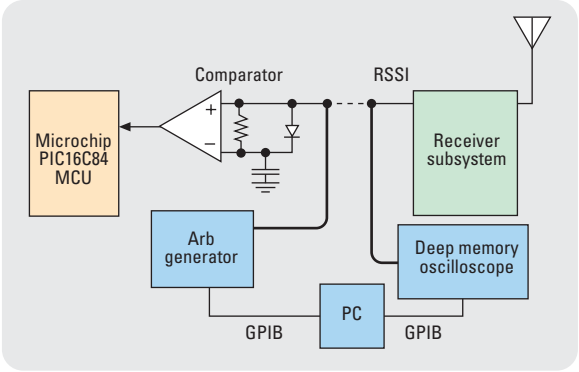


Figure 1. *A typical digital receiver circuit showing the RSSI signal that indicates the quality of the received signal.*

The second step is editing the captured waveform to introduce the sorts of error conditions we need to check for. PC connectivity software such as Agilent BenchLink or LabVIEW® makes it easy to transfer the scope data to a PC for editing. Since we're using the Agilent 33120A arbitrary waveform generator, the BenchLink Arb package offered a convenient way to do the editing. The editing tools let us mimic a variety of real-world conditions, including adding noise to simulate noisy transmission conditions, deleting data bits to simulate bit errors, and reducing the signal amplitude to simulate path loss.

The third step is downloading the modified waveform to the arbitrary waveform generator and injecting it into the circuit in place of the regular RSSI signal. We can then verify that the errors are detected and/or corrected, depending on the specific receiver. It's also easy to add noise in small increments until bit errors occur, which helps to characterize each model's sensitivity to noise. The arb generator also makes it possible to edit a specific bit, or set of bits, to make sure that errors in all positions are detected.

A sequence of waveforms will help demonstrate the process. Figure 2 shows the sort of RSSI signal we expect to get from the receiver subsystem. For comparison, Figure 3 shows the RSSI signal captured by the MSO, transferred to the PC and then recreated by the arbitrary waveform generator.

With the test system operational, we can now start modifying the waveforms to perform the what-if testing. In Figure 4, a noisy RSSI simulation exceeds the comparator threshold, thereby triggering a comparator output transition that is not present in the original digital signal. A test such as this helps us measure the sensitivity of the comparator in noisy environments.

Another common test we need to make is checking the receiver's response to missing, inadvertent or misplaced bits. In Figure 5, for instance, we inserted a bit error in the emulated RSSI signal.

The flexibility of arbitrary waveform generation means this kind of what-if testing is more or less limited only by your imagination and the nature of the circuit under test.

LabVIEW® is a U.S. registered trademark of National Instruments Corporation.

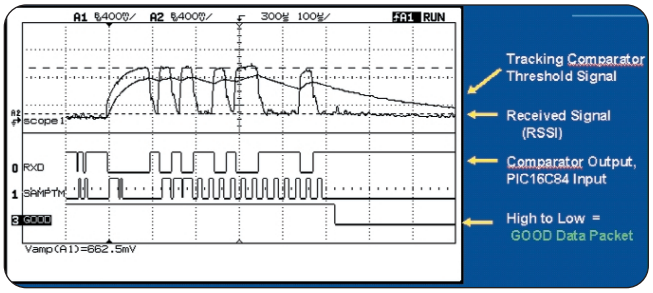


Figure 2. *A good RSSI signal and the corresponding comparator output that reconstructs the received serial data stream.*

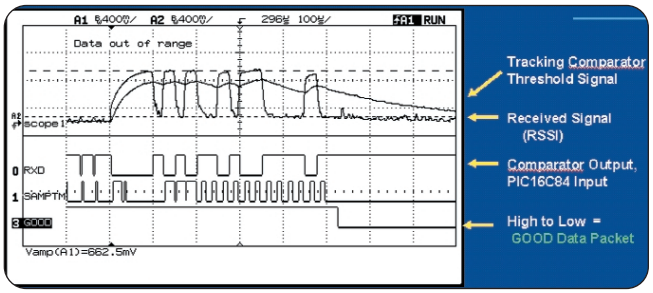


Figure 3. *The good RSSI signal recreated by arbitrary waveform generator.*

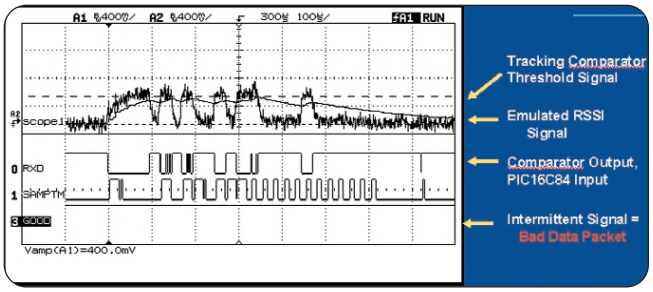


Figure 4. *A noisy signal exceeds the comparator threshold signal (near the right side of the display), generating a spurious comparator transition.*

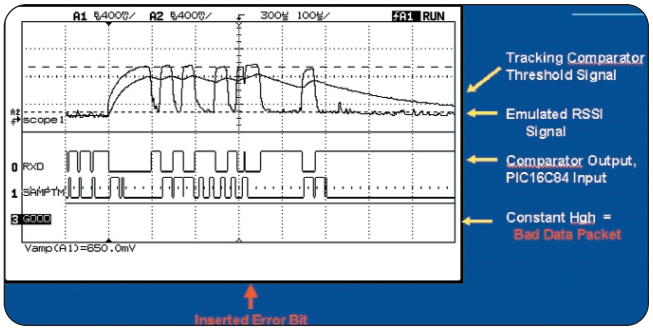


Figure 5. *A bit error inserted through the arbitrary waveform generator shows up as a bad data packet on the input of the MCU.*

Hint 6

Resolving hardware/software integration problems

By Charlie Howard, Embedded Technologies Associates, Inc.

One of the most common problems in MCU debugging is figuring out whether an anomaly is based in hardware or in software. This can be tricky enough if a single person designs both, and it's magnified many times over if a team of designers is involved.

Traditionally, hardware designers use an oscilloscope and logic analyzer to prove it's a software problem, while software designers use an emulator to prove it's a hardware problem. Unfortunately, these one-sided methods often only reconfirm that the problem exists. What we really need is a way to witness the problem as it occurs, while observing how the software and hardware behave and/or misbehave.

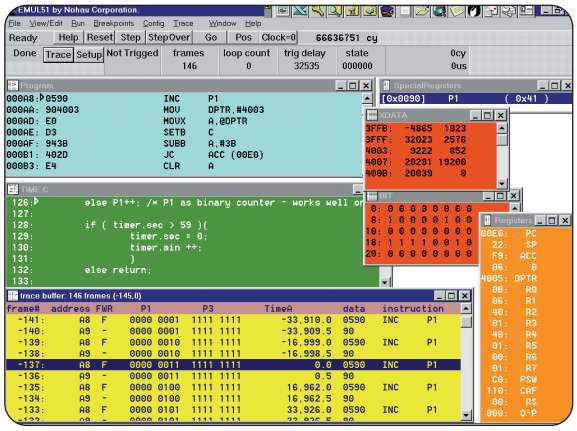


Figure 1. *The emulator's trace buffer shows the point (t=0) at which the emulator triggers the scope.*

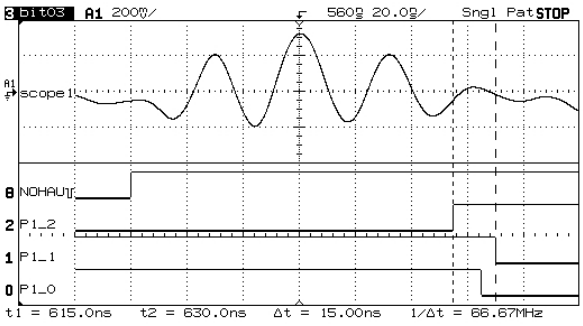


Figure 2. *A detailed look with the scope shows a signal delay on lines P1-1 and P1-0.*

Tying a logic analyzer to an emulator can help, but this can involve a lot of configuration and connection work. A faster, easier alternative that's more than adequate for most MCU-based designs uses an emulator's trace and triggering capability to trigger a mixed signal oscilloscope. At the same time, the emulator selectively stores the suspect software instructions.

In one recent debug scenario, I used a NOHAU 8031 emulator with a mixed signal scope to explore some trouble in the timing signals derived from the 8031 Port1 and an analog signal's relationship to these signals. Setup involved just three signals from the board to the emulator, one signal to the scope, and a trigger connection between the emulator and the scope.

As Figure 1 shows, the emulator captured the cycles in question while triggering the scope. (Note the time stamping.) The scope triggered on the fetch of the write to the port and captured the anomalous event (the slower transitions on lines P1-1 and P1-0) as well as the analog signal in question, indicating a hardware problem, as we can see from the scope display in Figure 2.

If this had been a software problem, I could've scrolled the trace buffer while synchronizing it to the source and program windows, making it easy to correlate program code to the error event. If there is more than one programmer writing to the same port, this method can save tremendous amounts of time and money by identifying the responsible software module. Plus, hardware engineers can continue to use the emulator's trigger-out and the second analog probe to isolate the cause of the problem further.

Correlating software and analog outputs in a CAN controller

By Pascal Mestdagh, EUROCORPS, Telecommunications Division

Hint

Until recently, troubleshooting mixed signal designs, where you need exact time coherency between analog signals and MCU code, was extremely difficult. The problem could be partially solved by combining a logic analyzer and an oscilloscope with common time bases and triggering them simultaneously. However, time base differences between the two instruments could lead to incorrect results. Moreover, differences in memory made things even more difficult. An alternative is to use a hybrid scope/logic analyzer. These instruments enhance cross-domain measurement accuracy and can reduce debugging time for mixed-signal designs.

In my application, where a Philips 80C51 MCU interacts with an 82C200 CAN (Controller Area Network) control chip to establish low-speed data communication between several domotics (home automation) devices, it is not always easy to determine the cause of an emerging problem. In this specific case, problems arose when I tried to send data to a remote device. It seemed as though several bytes were not arriving at their destination.

I connected the digital inputs of the scope to the MCU data bus and connected the scope's analog inputs to the transmission line (Figure 1). I then used pattern triggering to synchronize the measurement to the specific transmission request code word for the 82C200. Next, I set the trigger pattern in such a way that the measurement system triggered when the code word and the desired transmission frame occurred simultaneously. I quickly discovered that I had a software problem and had to review the code. Contrary to my first assumption, the test revealed that a data loss existed between the MCU and the CAN controller, and not on the transmission line (Figure 2).

The integrated scope and logic channels made it possible to compare with great accuracy the analog signals with their digital originators (the MCU code). In addition, deep memory is a big plus, since it let me sample the full length of the transmission frame (approximately 300 ms) and at the same time have enough detail to investigate the microcontroller code (approximately 150 ns).

Although conventional test equipment probably could've solved this problem, I saved a considerable amount of time using a hybrid analog-digital solution with deep memory.

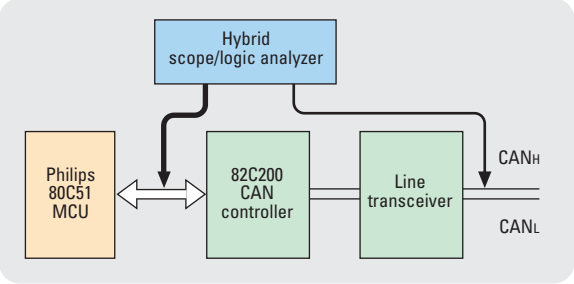


Figure 1. *Measurement connections used to debug the CAN controller setup.*

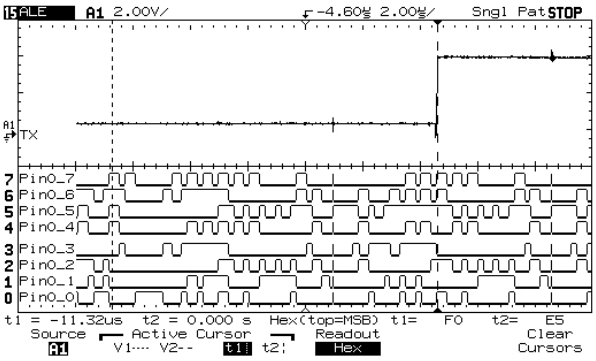


Figure 2. *The simultaneous occurrence of the transmission request code word and the analog transmission frame revealed an inconsistency in my software code. Bytes did not arrive at their destination because the MCU didn't verify the "transmission complete" bit of the status register in the CAN controller.*

Hint 8

Debugging an MCU-based CCD camera controller

by Jan Fischer, Petr Kocourek, and Petr Navratil

Like many MCU-based designs, the CCD camera systems we've been designing require simultaneous measurement of digital and analog signals, often with complex trigger requirements.

As Figure 1 shows, the horizontal synchronization pulses are first separated from the video signal. Using these Hsync pulses, the phase lock loop (PLL) generates the 10 MHz ADCLK signal. The falling edge of ADCLK samples the TV signal into the ADC; its rising edge updates the ADC output. The programmable logic device (PLD) converts ADCLK to generate the $\overline{WR}$ signal. The $\overline{WR}$ rising edge writes the data from the ADC into the FIFO memory, which then contains a digitized signal of one TV line. The Philips 80C552 MCU reads data from the FIFO and calculates the feedback control data for camera positioning and zooming. Systems like this are commonly used in applications that need to track and measure objects visually, such as navigation and noncontact measurement.

Using the single-shot TV trigger mode and Autostore function of our mixed signal scope (MSO), we discovered a 25 ns edge instability or jitter on the ADCLK signal. With the MSO, we easily captured and stored a complete 20 ms stream of one-half of the TV picture at 50 MSa/s for further processing and analysis (Figure 2).

Figure 3 shows the critical timing of writing the data from ADC into the memory; 5 ns was not enough time for the memory to store the data. This problem, which was impossible to find with a conventional scope, was obvious when we measured with a hybrid scope/logic analyzer. Using these results, we were able to reprogram the PLD to avoid the problem.

The ability to trigger the MSO with a TV signal made it easier to debug the MCU software routine. The combination of analog and digital acquisition gave us a complete view of some rather complex behavior in our design.

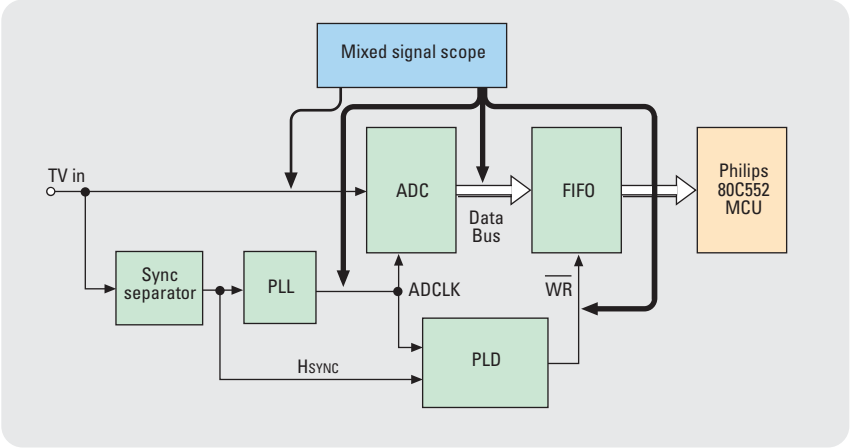


Figure 1 *Block diagram of the CCD camera controller showing analog and digital test connections.*

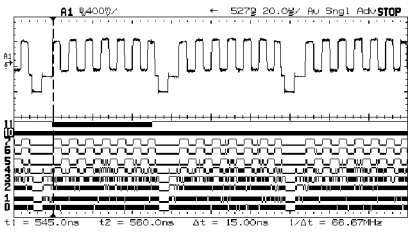


Figure 2. *The analog output of the CCD camera and the relevant digital signals in the control system. The output of the ADC is on lines 0-7, the ADCLK signal is on line 10, and the $\overline{WR}$ signal is on line 11.*

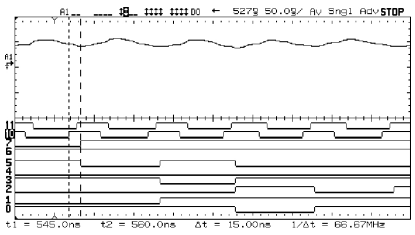


Figure 3. *The display markers highlight the timing relationship between ADCLK and $\overline{WR}$.*

Jan Fischer and Petr Kocourek are with Czech Technical University and Petr Navratil is with T&M Direct.

Rock-solid waveforms at rock-bottom prices

Tackle your test and verification tasks using the clean, stable waveforms built into these function/arbitrary waveform generators from Agilent — and when a standard signal isn't enough, create your own custom arbitrary waveforms to simulate real-world signals.

Both the Agilent 33120A and the 33250A function/arbitrary waveform generators offer a lot of capability and performance for a very affordable price. Standard outputs include sine, square, ramp, noise, sin(x)/x, dc volts and more. AM, FM and FSK capabilities make it easy to modulate waveforms with or without a separate source.

With the 33250A, you can also generate simple pulses up to 50 MHz. And the color graphical display and user-friendly front panel make complicated tasks easy to accomplish.

For system applications, both GPIB and RS-232 interfaces are standard, and support full programmability using SCPI commands.

Choose the performance that's right for your application

Agilent 33120A

- 15 MHz sine and square wave outputs
- Build arbitrary waveforms with 40 MSa/s speed and storage for four 16,000-point waveforms

Agilent 33250A

- 80 MHz sine and square wave outputs
- Build arbitrary waveforms with 200 MSa/s speed and storage for four 64K-point waveforms
- 50 MHz pulse waveforms with variable rise/fall times

	33120A	33250A
Waveforms		
Standard	Sine, square, ramp, triangle, noise, sin(x)/x, exponential rise and fall, cardiac, dc volts	Sine, square, ramp, pulse, noise, sin(x)/x, exponential rise and fall, cardiac, dc volts
Arbitrary		
Waveform length	8 to 16,000 points	1 to 64 K points
Nonvolatile memory	4 waveforms (each from 8 to 16,000 pts)	4 waveforms (each from 1 to 64K pts)
Vertical resolution	12 bits	12 bits
Sample rate	40 MSa/s	200 MSa/s
Frequency characteristics		
Sine/Square	1 µHz to 15 MHz	1 µHz to 80 MHz
Ramp/ Triangle	100 µHz to 100 kHz	1 µHz to 1 MHz
Pulse		500 µHz to 50 MHz
White noise	10 MHz bandwidth	50 MHz bandwidth
Resolution	10 µHz or 10 digits	1 µHz; except pulse, 5 digits
Accuracy	10 ppm in 90 days (18 °C to 28 °C)	0.3 ppm (18 °C to 28 °C)
THD (dc to 20 kHz)	0.04%	<0.2% + 1 mVrms
Output characteristics		
Amplitude		
Into 50 Ω	50 mVp-p to 10 Vp-p	10 mVp-p to 10 Vp-p
Accuracy (at 1 kHz)	±1% of specified output	±1% of setting ±1 mVp-p
Flatness (sine wave relative to 1 kHz, Autorange,into 50 Ω)	<100 kHz ±1% (0.1 dB) 100 kHz to 1 MHz ±1.5% (0.15 dB) 1 MHz to 15 MHz Ampl >3Vrms ±2% (0.2 dB) Ampl <3Vrms ±3.5% (0.3 dB)	<10 MHz ±1% (0.1 dB) 10 MHz to 50 MHz ±2% (0.2 dB) 50 MHz to 80 MHz ±5% (0.4 dB)
Modulation		
AM		
Modulation	Any internal waveform, including arb	Any internal waveform, including arb
Frequency	10 mHz to 20 kHz	2 mHz to 20 kHz
Source	Internal/external	Internal/external
Depth	0% to 120%	0% to 120%
FSK		
Internal rate	10 mHz to 50 kHz	2 mHz to 1 MHz
Frequency range	10 mHz to 15 MHz	1 µHz to 80 MHz
FM		
Modulation	Any internal waveform, including arb	Any internal waveform, including arb
Frequency	10 mHz to 10 kHz	2 mHz to 20 kHz
Deviation	10 mHz to 15 MHz	dc to 80 MHz
Source	Internal only	Internal/external
Sweep		
Type	Linear or logarithmic	Linear or logarithmic
Start/Stop Frequency	10 mHz to 15 MHz	100 uHz to 80 MHz
Sweep Time	1 ms to 500 s	1 ms to 500 s
Trigger	single, external, or internal	Single, external, or internal
Marker		Falling edge of sync (programmable)
Burst		
Waveform frequency	5 MHz max.	1 µHz to 80 MHz
Count	1 to 50,000 or ∞ cycles	1 to 1,000,000 or ∞ cycles
Start/Stop phase	–360.0° to +360.0°	–360.0° to +360.0°
Internal period	10 mHz to 50 kHz	1 µs to 500 s
Gate source	Internal/external	External
Trigger source	Single, external, or internal	Single manual trigger, internal, or external
Warranty	**3 years**	**3 years**

∞ = *infinity symbol*

See an interactive product overview at
http://www.agilent.com/find/waveform

33120A

33250A

Easily see what's happening in your mixed analog and digital designs

Agilent 54600 Series scopes are optimized with just the capabilities you need for verifying and debugging designs that include embedded 8- or 16-bit microcontrollers:

- 2 MB MegaZoom deep memory on each channel so you can capture long, non-repeating signals, maintain high sample rate and quickly zoom in on areas of interest;
- a revolutionary ultra-responsive, high-definition display that's a clearer "window into your world"—it lets you see more signal detail than ever before;
- flexible triggering that lets you easily isolate and analyze the complex signals and fault conditions common in mixed analog and digital designs.

Agilent 54600 Series scopes

Multiple configurations to meet your needs

Mixed signal scopes: With 2 analog channels and 16 digital channels, these scopes combine the detailed signal analysis of a scope with the multichannel timing measurements of a logic analyzer, so you can simultaneously test and monitor the high-speed digital control signals and the slower analog signals in your design.

4-channel scope: If your designs include heavy analog content, the 100-MHz 54624A will give you the channel count and measurement power you need to get your debug and verification done with ease.

2-channel scopes: The 2-channel scopes give you an affordable way to see long time periods while maintaining high sample rate so you can see details in your designs.

See for yourself with a free demo

Every scope user knows that the real test is how well the instrument performs in your environment, with your design. Call Agilent Technologies to arrange a free demo.

You can also check out the benefits of MegaZoom at **www.agilent.com/find/MegaZoom.**

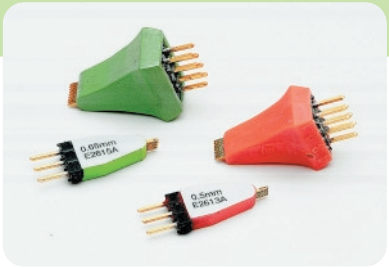
	54621A	54621D	54622A	54622D	54624A
Channels	2	2+16 logic	2	2+16 logic	4
Scope					
Bandwidth	60 MHz	60 MHz	100 MHz	100 MHz	100 MHz
Sample rate	200 MSa/s				
Memory depth	2 MB/ch MEGAZoom				
Logic					
Max sample rate	n/a	400 MSa/s	n/a	400 MSa/s	n/a
Max memory depth	n/a	4 MB/ch	n/a	4 MB/ch	n/a
Display	High definition with 32 levels gray scale; 1,000-point horizontal resolution				
Display update rate	Up to 25,000,000 vectors/sec per channel				
Timebase (per division)	5 ns to 50 s				
Triggering	Edge, pulse width, pattern, I²C, TV, sequence, duration				
Peak detect	5 ns				
Measurements	Peak, average, RMS, max, min, frequency, period, pulse width, rise/fall time, duty cycle				
Waveform math	Subtraction, multiplication, FFT, integration, differentiation				
Storage	Built-in 1.4 M floppy disk				
Connectivity	RS-232, parallel standard; optional GPIB; optional integrated printer				
Built-in help	In 9 languages				
Warranty	3 years standard, optional increase to 5 years				

Dependable, trouble free connection to fine-pitch ICs

The Agilent Wedge probe tip adapter solves the problem of connecting your scope or logic analyzer to fine-pitch, TQFP and PQFP surface-mount ICs. It works by inserting compressible dual conductors between adjacent IC legs. The flexible conductors conform to the size and shape of each leg to ensure tight contact. It's then a simple matter to connect your scope or logic analyzer to the Wedge.

The Wedge's unique design delivers secure, redundant contact on each leg, with little chance of shorting to adjacent legs. Plus, it's mechanically non-invasive, so it won't damage your device under test.

Agilent Wedges are available with 3-, 8- and 16-leg connections for 0.5 and 0.65 mm IC packages.



The Agilent Wedge provides dependable, trouble-free connection to fine-pitch ICs.

Affordable and user friendly logic analysis

The Agilent LogicWave PC-based logic analyzer will have you making measurements in just minutes. With its familiar Windows interface, LogicWave is easy to set up and use, yet it still offers the speed and performance you need for serious logic analysis.

LogicWave features 34 channels of intuitive logic analysis (32 state channels), 100 MHz state analysis, and 250 MHz timing analysis, with memory depths up to 128 K. Reliability, signal fidelity and other quality measures far exceed the typical PC-based logic analyzer, too.

And LogicWave is priced low enough that it's the ideal personal analyzer for design teams who currently share a primary logic analyzer.

Experience this new level of simplicity yourself by downloading the free user interface software from the LogicWave web site: **www.agilent.com/find/LogicWave** or call Agilent and ask for the the software on CD-ROM.

Model	Agilent Technologies LogicWave (E9340A)
State/timing channels	34
Maximum state clock	100 MHz
Maximum timing sample-rate	250 MHz
Memory depth	128K timing, 64K state
User interface	Windows 95/98/NT, PC-hosted (runs as an application on any Pentium or better, desktop or laptop)
"WYDIWYC"timing trigger	"What you draw is what you capture" visual timing trigger events
Printers	Shared with the host PC — can print to any local or network printer supported by the PC
Probing	Agilent patented, 100 kΩ, 1.5 pF
Dimensions	11.5" x 9" x 2.5" (29.1 x 22.8 x 6.3 cm)
Weight	4.5 pounds (2.1 kg)
I/O ports	Enhanced Parallel connection to PC for fast deplay update rates, trigger IN/OUT BNC

www.agilent.com/find/gp

Get FREE Agilent test equipment. See page 2 for details.

Agilent Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive and minimize your risk and problems. We can help you choose the right products and apply them successfully. Every instrument/system we sell has a global warranty and is supported at least five years beyond its production life. Two concepts underlay the Agilent support policy: "Our Promise" and "Your Advantage."

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. We can verify that it works properly, help with product operation, and provide basic measurement assistance for specified capabilities, at no extra cost. Many self-help tools are available.

Your Advantage means that Agilent offers additional expert test and measurement services at extra cost. Solve problems efficiently by contracting us for calibrations, extra-cost upgrades, out-of-warranty repairs, on-site training, professional engineering services, and more.

By internet, phone, or fax, get assistance with all your test & measurement needs

Online assistance: **www.agilent.com/find/assist**

Phone or Fax

United States:
(tel) 1 800 452 4844

Canada:
(tel) 1 877 894 4414
(fax) (905) 206 4120

Europe:
(tel) (31 20) 547 2000

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Latin America:
(tel) (305) 267 4245
(fax) (305) 267 4286

Australia:
(tel) 1 800 629 485
(fax) (61 3) 9272 0749

New Zealand:
(tel) 0 800 738 378
(fax) 64 4 495 8950

Asia Pacific:
(tel) (852) 3197 7777
(fax) (852) 2506 9284

Product specifications and descriptions in this document subject to change without notice.

Printed in USA 4/00
5980-0943EN

Agilent Technologies
Innovating the HP Way