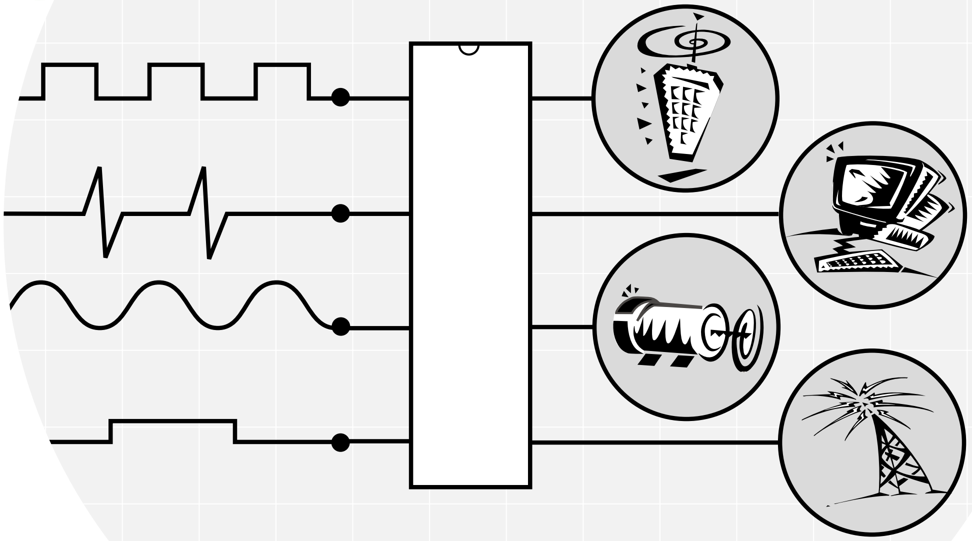
5 Hints for Debugging Microcontroller-based Designs

Application Note 1458

Time-saving tips from successful designers using Microchip *based systems.*



Microchip

Agilent Technologies

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 (MCU). 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 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.

Verifying quadrature encoder signals in a motion control system

Hint 1

By Lon Glazner, Solutions-Cubed

Many control algorithms in motion control systems require position and/or velocity feedback. This feedback can come from many sources, including analog signals, absolute position encoders, and quadrature encoders. This hint refers to quadrature encoders, which offer a low-cost feedback solution with excellent resistance to environmental changes. Quadrature encoders typically provide magnetic or optical outputs consisting of two channels, A and B, and sometimes an index output related to a specific motor position.

Controlling motor speed, direction, and shaft position

In a typical application, a quadrature encoder provides TTL level outputs that relate to motor speed and direction. Motor speed is determined by the rate at which encoder pulses are generated and is based on a variety of factors. These factors include the counts-per-revolution (CPR) of the quadrature encoder and the gear ratio of the motor. Many quadrature encoders are mounted on the rear of the motor prior to any motor gearing, while others are mounted on the motor load. If motor speed is the only desired feedback signal, then either the A or B channel of the quadrature encoder can be used, both channels are not needed.

Monitoring both channels A and B and determining their phase relationship to each other determines motor direction. Figure 1 shows that a motor moving in the forward direction generates a pulse on channel A that leads the pulse output from channel B. Quadrature encoders also have a CPR rating that can be associated with shaft position. For example, a 500CPR encoder on a motor with no gearing generates 1000 pulses (or counts) with two complete motor shaft revolutions.

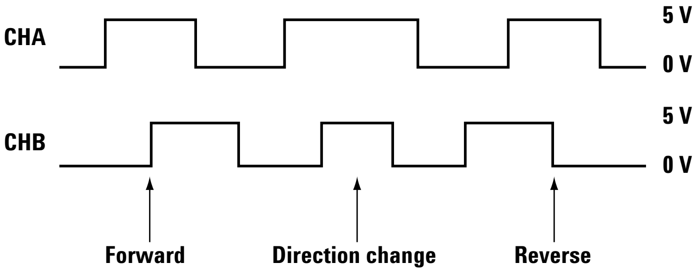


Figure 1. The phase relationship of encoder channels A and B determines motor direction. A motor moving in the forward direction generates a pulse on channel A that leads the pulse output from channel B.

Hint 1

Converting quadrature encoder output

Whenever channel A leads channel B, we will call it an “up-count.” Conversely, when channel B leads channel A, we will call it a “down-count.” By summing the up-counts and down-counts, you can generate a number associated with the distance a motor has moved in the positive or negative direction. Converting the quadrature output channels to up-counts and down-counts can be done with discrete logic circuits, in software, or with specialized integrated chips designed for this purpose.

For this hint, we will use an LS7083 from LSI Computer Systems, Inc. to convert the quadrature encoder channels to up-counts and down-counts. Additionally, we will use a PIC18F252 microcontroller from Microchip Technology, Inc. to convert the up-counts and down-counts to position and velocity values (see Figure 2). Though not shown on the schematic, the PIC18F252 also generates control signals for an external H-bridge driving a brushed DC motor. The PIC18F252 is configured to increment its internal Timer 0 and Timer 1 counters upon external signal transitions. Additionally, a separate timer (Timer 2) associated with the pulse-width-modulation update rate of the microcontroller serves as a time base for velocity values.

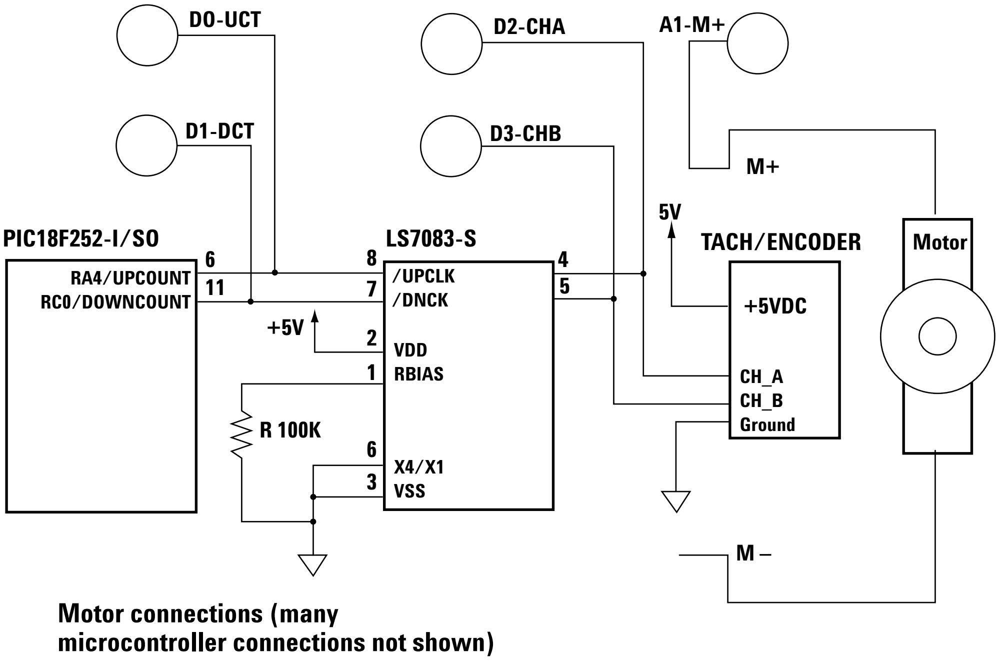


Figure 2. Schematic of a PIC18F252 microcontroller that converts up-counts and down-counts to position and velocity values.

Here is an example of simple C code for generating position and velocity values within an interrupt subroutine. Note that velocity is maintained as a positive or negative value when the signal denotes the direction of motor travel.

```
If (PIR1bits.TMR2IF)                        //Test for PWM update interrupt
{
    PIR1bits.TMR2IF = false;                //Clear interrupt flag
    Velocity = DownCount;                   //Subtract out old counter values
    Velocity - = UpCount;
    UpCount = ReadTimer0();                 //Load upcount
    DownCount = ReadTimer1();               //Load downcount
    Velocity += UpCount;                    //Velocity = difference of counters
    Velocity - = DownCount;
    Position += Velocity;                   //Position = sum of velocity values
}
```

Verifying analog and digital feedback signals

Serious consideration should be given to verification of the feedback signals. The mounting techniques of quadrature encoders can differ between motor manufacturers, and their output signals should be verified. After-market quadrature encoders mounted on motor rear shafts or motor loads can cause unforeseen problems. The most common problem occurs when a down-count is generated while the motor is turning in the forward direction. This can cause a runaway condition in a position control system.

Using an Agilent 54642D mixed signal oscilloscope (MSO), the signals levels of the quadrature encoder, the LS7083 converter chip, and the analog signal of the motor's positive lead can all be monitored at the same time. In fact, without the additional digital channels available on a 54642D MSO, multiple oscilloscopes, or a digital analyzer, and complex external triggering would be required to perform this simple verification process.

To verify that your quadrature encoder counts up when the motor is running forward, simply connect the 54642D MSO to the test points shown in Figure 2 and set the trigger for the rising edge of D2-CHA. As in Figure 3, you should see that D2-CHA leads the signal at D3-CHB, and that an up-count is present (a short logic 0) on D0-UCT. If this is not the case, you can remedy the problem by swapping your CHA and CHB connections.

Many 2- or 4-channel oscilloscopes can monitor the analog signals related to brushed DC motor control, but it is common for a variety of digital and analog signals to be present in a motor control design. These signals could be monitored with multiple instruments, however a better solution is available. A 54642D MSO provides a variety of analog and digital measurement capabilities, which makes it ideal for use in designing and verifying mixed-signal motor control systems.

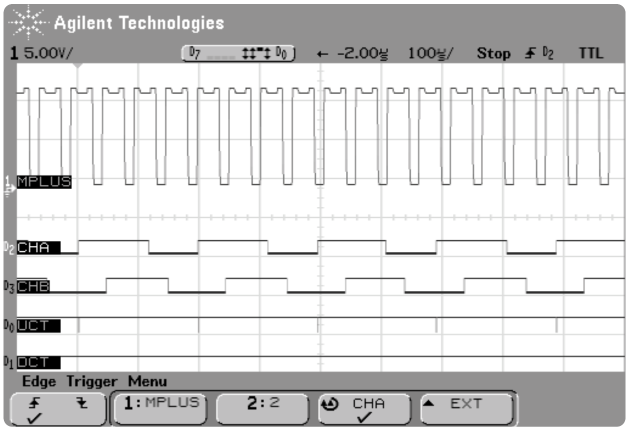


Figure 3. To verify that the encoder counts up when the motor is running forward, connect the 54642D MSO to the test points shown in Figure 2 and set the trigger for the rising edge of D2-CHA. You should see that D2-CHA leads the signal at D3-CHB, and that an up-count is present (a short logic 0) on D0-UCT.

Hint 2

Implementing a simple voltage follower using a digital pentiometer

By Frank Rossini, Solutions-Cubed

Digital potentiometers or pots have many uses in today's embedded systems. In this example, we will implement an embedded "voltage follower" using a PIC16F873 microcontroller and a MCP41010 digital pot, both from Microchip Technology, Inc. Basically, the PIC microcontroller (MCU) will read the analog voltage and instruct the digital pot to reproduce the input voltage. Because we are interested in analyzing the analog input and output and the smart plug-in interface (SPI) to the digital pot, the mixed-signal analysis capabilities of the Agilent 54642D mixed signal oscilloscope (MSO) will come in handy.

Designing the voltage follower

Figure 1 shows the simplified system used for testing, which consists of a filtered analog input to a PIC16F873, three digital lines connecting the PIC MCU to the MCP41010 pot, and the output of the pot. Two analog and three digital lines on the oscilloscope monitor the system.

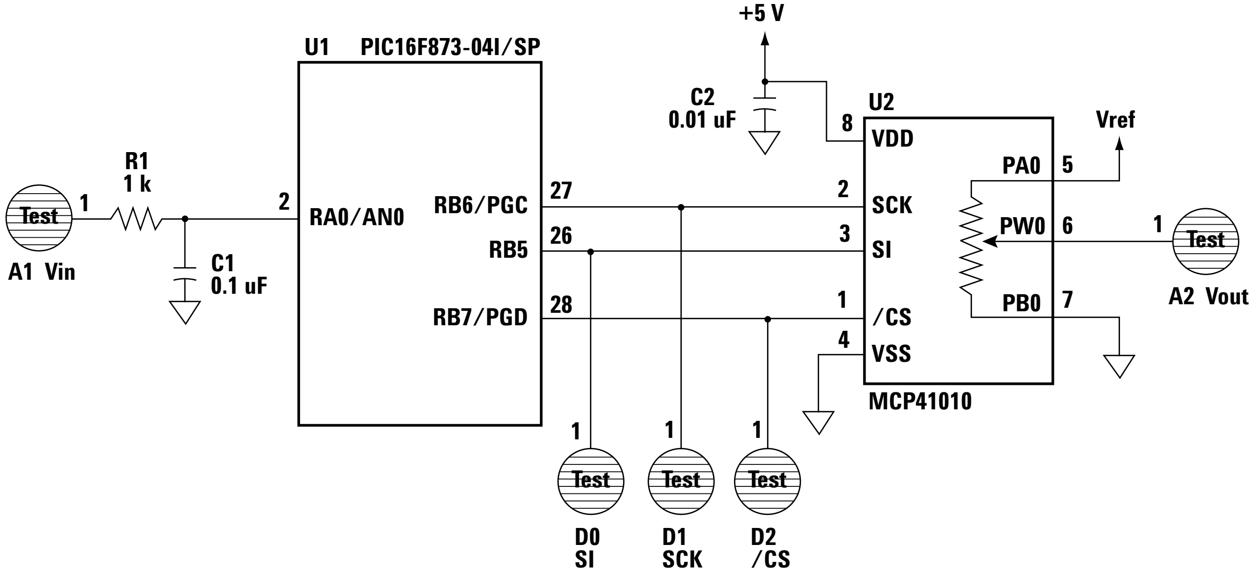


Figure 1. Simplified system diagram showing the filtered analog input to a PIC16F873, three digital lines connecting the PIC MCU to the MCP41010 digital pot, and the output of the pot. Two analog lines and three digital lines on the 54642D mixed signal oscilloscope monitor the system.

Hint 2

This abbreviated C code reads the input voltage on the Channel 0 A/D of the PIC MCU, converts it to an 8-bit value, and sends the result directly to the digital pot via the SPI. This allows the designer to input various voltages and easily test the potentiometer output. The format of the data sent to the pot is beyond the scope of this document and is available from Microchip if desired.

```
int8 Get_Voltage(int8 Channel, int8 Count)
{
    set_adc_channel(Channel);
    delay_us(200);                              //Sample & Hold Time
    Vavg = 0;
    for(x=0;x<Count;x++)                        // Take "Count" Samples
    {
        delay_us(50);                           // Sample & Hold Time
        Vavg = Vavg + (read_adc() >> 2);        // Use only 8 bits out of 10
    }
    Vavg = Vavg / Count;                        // Get Average
    return (int8)(Vavg);
}

void Digital_Pot_Control(int8 Pot_Output)
{
    output_high(CS_41010);                      //Start with CS line high
    output_low(DAT_41010);                      //Start with control lines low
    output_low(CLK_41010);                      //
    delay_cycles(2);                            //Small Delay
    output_low(CS_41010);                       //Assert Chip Select

    //Control Byte Loop - 8 bit constant
    Pot_Temp = 0x11;                            //Value = 00010001 (Write to Pot0)
    for(x=1;x<9;x++)                            // Send 8 bits
    {
        if( bit_test(Pot_Temp,7) == 1 )         //Test for one or zero
            output_high(DAT_41010);
        else
            output_low(DAT_41010);
        shift_left(&Pot_Temp,1,0);
        output_high(CLK_41010);                 //Clock in Data
        delay_cycles(2);                        //Small Delay
        output_low(CLK_41010);                  //
        delay_cycles(2);                        //Small Delay
    }

    //Data Byte Loop - 8 bit constant
    Pot_Temp = Pot_Output;
    for(x=1;x<9;x++)                            //Send 8 bits
    {
        if( bit_test(Pot_Temp,7) == 1 )         //Test for one or zero
            output_high(DAT_41010);
        else
            output_low(DAT_41010);
        shift_left(&Pot_Temp,1,0);
        output_high(CLK_41010);                 //Clock in Data
        delay_cycles(2);                        //Small Delay
        output_low(CLK_41010);                  //
        delay_cycles(2);                        //Small Delay
    }
    output_low(DAT_41010);                      //
    output_high(CS_41010);                      //Unassert CS line
}

main()
{
    while(TRUE)                                 //Main Program Loop Begin
    {
        restart_wdt();                          //Reset Watchdog Timer
        delay_ms(1);
        Simulated_TPS = Get_Voltage(Channel0, 1); //Take 1 Sample
        Digital_Pot_Control(Simulated_TPS);
    }                                           //End of Main Program Loop
}                                               //End of main()
```

Hint 2

Analyzing the analog input and digital output

The top analog trace in Figure 2a represents the input voltage, the analog voltage reading at the bottom of the figure represents the digital output for the SPI. Notice how the output voltage changes after the /CS line is unasserted on the SPI bus. It is also worth noting that the 54642D MSO has built-in SPI triggering. You can select the lines to use for CS, Clock, and Data, pick between rising and falling edge clocked data, and even select the value of the data byte to trigger on.

The sine-wave screen shown in Figure 3 displays the flexibility of the 54642D. Using deep memory, a feature that makes the MSO very easy to use, the time base can be expanded to read the specific SPI data for each analog section. In addition to SPI triggering, the MSO has triggering features for USB, I^{2}C, and CAN Bus. Other common measurements, such as phase delay and frequency, are also easily displayed. Only three digital channels were used in this example; by using the 13 additional digital inputs of the oscilloscope, more data can be viewed.

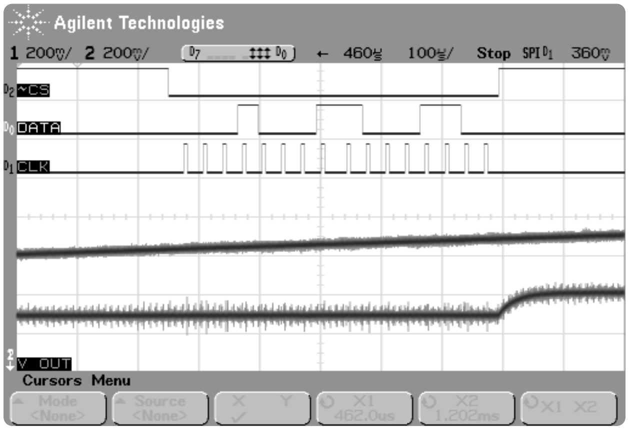


Figure 2a. Measurement of the test system showing the input voltage (analog trace on top) and the digital output (analog voltage on the bottom).

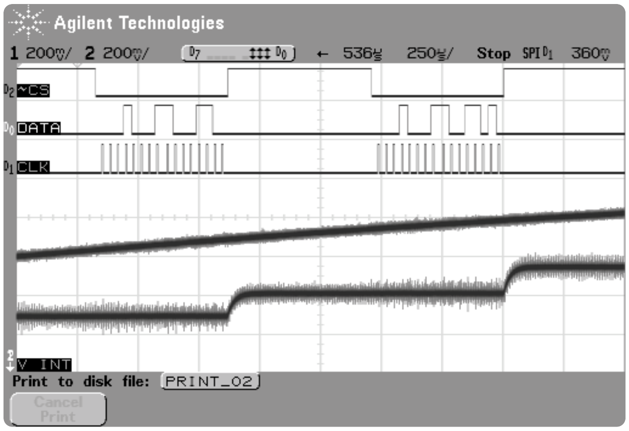


Figure 2b. Another view of the test system with two transitions.

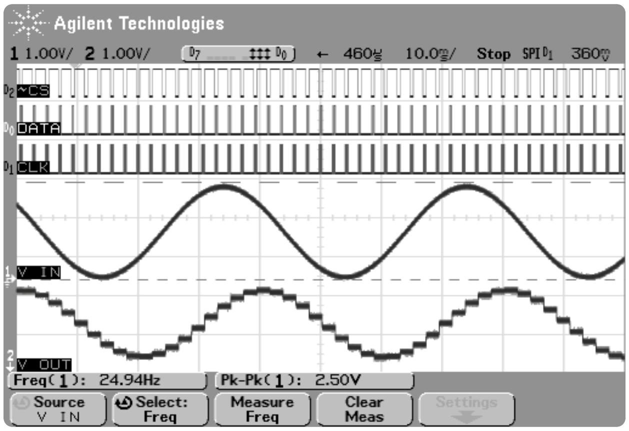


Figure 3. A sine wave screen showing the digital pot following an analog input.

Monitoring a digital motor speed controller

By Lon Glazner, Solutions-Cubed

3 Hint

Verifying motor control signals with instrumentation presents various challenges to the designer. For example, a digital motor controller can require multiple digital and analog test points to verify functionality throughout a design. If you couple this with the need to monitor motor voltage and current, as well as digital drive signals, the challenges can become overwhelming for companies with small instrumentation budgets. Fortunately, the Agilent 54642D mixed signal oscilloscope (MSO) coupled with the Agilent 1146A AC/DC current probe offer these capabilities at a reasonable price. This hint will describe some of the components of a digital speed controller for a brushed DC motor and finish with examples of signal verification.

Implementing the digital motor control

Many low cost, off-the-shelf components are available to control the speed and direction of brushed DC motors. Speed is controlled by monitoring the average voltage seen across the DC motor or other similar load. This is most commonly done with pulse-width-modulation (PWM) techniques. A PWM signal can be described as a periodic signal with an adjustable “on” time. This “on” time is referred to as the positive duty-cycle. Most motor control applications require a PWM signal with a frequency in the 15 to 30 kHz range.

Many microcontrollers have built-in PWM generating hardware and offer highly versatile platforms for generating motor control signals. For this hint we will be using a PIC16F873 from Microchip Technology, Inc., which offers two PWM output channels.

The PIC16F873 outputs logic level (0 to 5 V) signals that are not appropriate for directly connecting to brushed DC motors. Therefore, a method of “amplifying” these PWM signals is required to interface to, and control, the motor itself. A common configuration of discrete transistors, the H-bridge, often provides this digital-to-analog interface and a method to control motor direction as well as speed. The H-bridge is composed of two high-side transistors attached between the motor and the motor voltage and two low-side transistors that connect the motor to ground. In this hint, a low-ohm sense resistor is used to monitor the current.

Hint 3

The H-bridge earns its name from its schematic representation. Four transistors acting as switches direct the current path through the motor. Figure 1 shows the logic state of the transistors (1 = on, 0 = off) and the manner that they are turned on or off to control motor direction. N-channel MOSFETs are the ideal transistor switches for low-voltage, medium-current applications. They can switch on and off rapidly and they present low-resistance current paths when turned fully on. The Intersil HRF3205S is a 55 V N-channel MOSFET that can carry up to 10 A. By applying PWM signals to the low-side transistors, the average motor voltage can be adjusted. For example, in a 24 V system a 50 percent positive duty-cycle would apply 24 V across the motor 50 percent of the time. Therefore, the average motor voltage would be 12 V, and the motor would operate at half speed. For accurate speed control, feedback from a tachometer or encoder would be required to account for changes in the motor load or voltage.

Building an H-bridge out of N-channel MOSFETs requires level translation circuitry. This circuitry provides gate-drive currents and voltages that make it possible to use N-channel MOSFETs for both the low-side and high-side transistors. One such integrated circuit is the Intersil HIP4081, which allows direct control of each MOSFET making up the H-bridge.

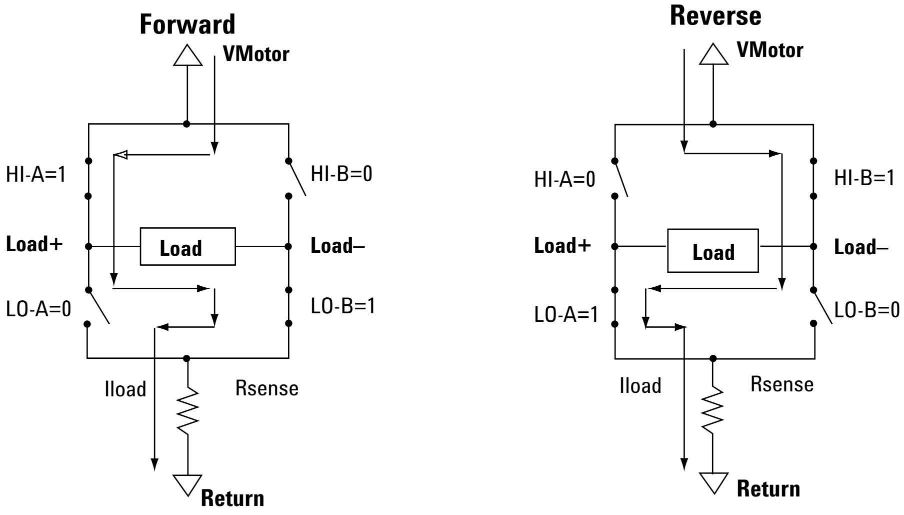


Figure 1. Logic state of the H-bridge transistors (1=on, 0=off) and the manner for turning them on and off to control motor direction.

Hint 3

Figure 2 shows a block diagram of the connections between the microcontroller, level translation circuit, and the four MOSFETs that make up the H-bridge. This bare-bones diagram does not include many of the support components required for an actual motor control circuit.

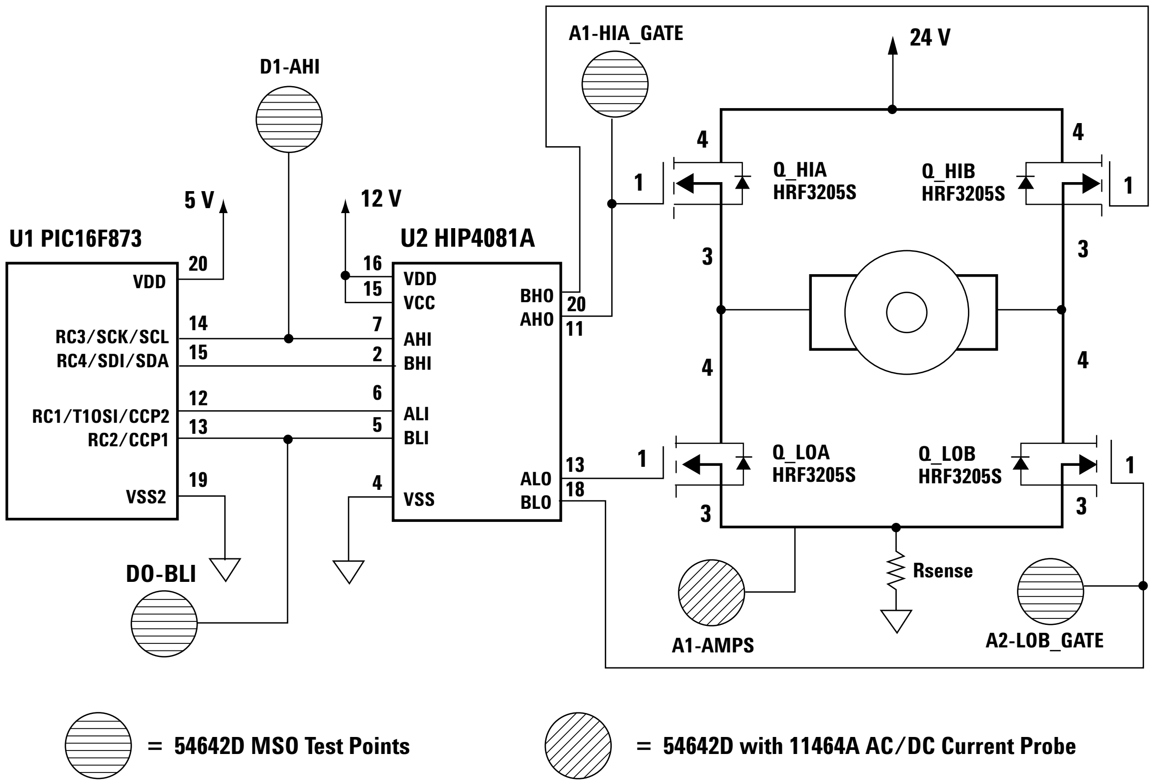


Figure 2. Basic diagram of the connections between the microcontroller, the level translation circuitry, and the four MOSFETS of the H-bridge.

Hint 3

Monitoring the analog and digital signals

Using a 54642D MSO, the motor control analog and digital signals can be measured. Assuming the system is running in the forward direction, you would expect to see a digital PWM signal at D0-BLI and a digital "1" at D1-AHI. These control signals generate gate-drive signals for the MOSFETs. The PWM signal measured at A2-LOB_GATE will be 0 V for "off" and 12 V during the positive duty-cycle (the actual supply for this system is closer to 11.3 V). The high-side MOSFET (Q_HIA) needs to have its gate driven at a voltage higher than its source. When Q_HIA is "on," its source provides 24 V to the positive side of the motor. Therefore, we expect to see the signal at A1-HIA_GATE to be roughly 12 V higher than the source, or 36 V. By triggering on the rising edge of D0, you can measure propagation delays and signal rise times for the PWM signal, and verify the gate-drive voltage of the high-side transistor. Using the "Quick Meas" menu in the 54642D MSO, the gate-drive voltage rise time can be quickly discerned.

Many motor control systems require over-current protection circuitry. It is very difficult to verify circuit accuracy in a non-intrusive manner without a current probe. The Agilent 11464A AC/DC current probe allows the designer to take these measurements without interrupting the motor connections, and can display the information on the 54642D MSO.

In Figure 4, a resistive load is attached in place of a motor to test the current handling capability of the motor controller. With the positive duty-cycle set to 80 percent, the load is adjusted so that an average current of roughly 5 A is applied to the load. With an average current of 5 A, ideally you would see 6.25 A for 80 percent of the time and 0 A for 20 percent of the time.

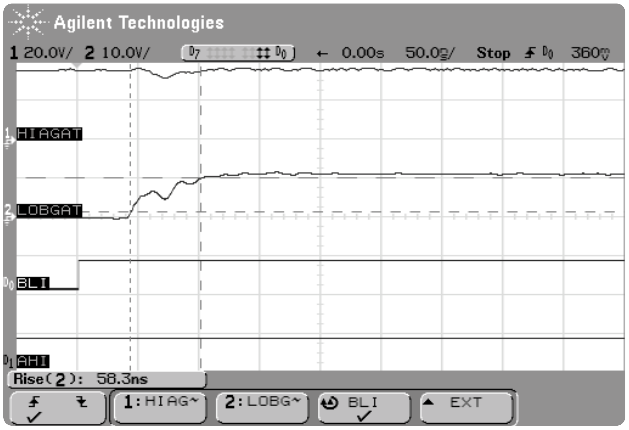


Figure 3. Using the "Quick Meas" menu in the 54642D MSO, the gate-drive voltage rise time can be quickly discerned.

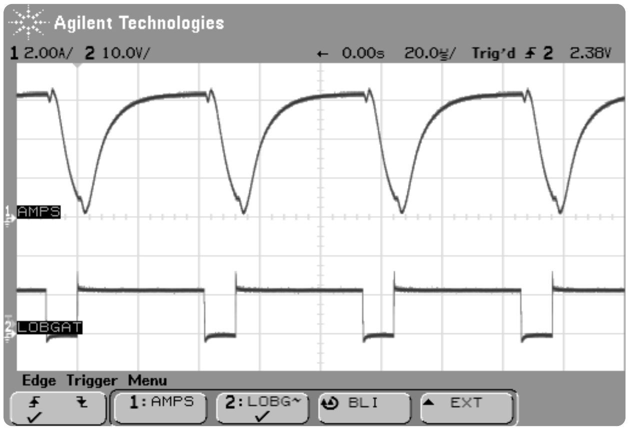


Figure 4. A measurement showing the motor current in relation to the low-side (Q_LOB) MOSFET PWM control signal. With the 11464A AC/DC current probe set for 100 mV/Amp, the signals displayed show the expected current waveform. The 54642D MSO is set to trigger on the rising edge of the gate-drive signal of the MOSFETs.

Figure 4 shows the motor current in relation to the low-side (Q_LOB) MOSFET PWM control signal. With the 11464A set for 100 mV/Amp, the signals displayed show the expected current waveform. The 54642D MSO is set to trigger on the rising edge of the gate-drive signal of the MOSFETs.

Additional verification using 16 digital channels

In this design, we concentrated on taking measurements of a speed control system running in one direction only. Therefore, only two MOSFET drive signals are displayed in Figure 3. With the number of digital channels available on the 54642D MSO (16), it is feasible to monitor all four MOSFET drive signals, which would be useful for bi-directional systems. An additional 12 digital level signals, including limit switches, brake inputs, and other digital control or feedback signals, could be monitored. The ability to monitor so many digital channels enhances the prospects of completing a digital speed-control design successfully.

Motor controllers designed today have a multitude of analog and digital signals that must be verified during system design. It is often difficult to troubleshoot problems and verify signal integrity with a minimum of instrumentation channels. With the 2 analog and 16 digital channels available on a 54642D MSO, the interrelationship of the system input and output signals can be quickly determined. By adding accessories such as the 11464A AC/DC current probe, the usefulness of the oscilloscope is increased without significant cost. These two Agilent tools are a must for mixed-signal design engineers and their firms.

4 Hint

Basic tilt sensing using a MEMS accelerometer

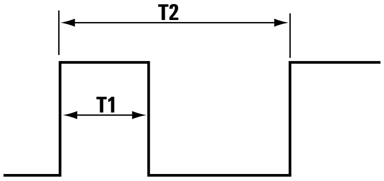
By Frank Rossini, Solutions-Cubed

Micro-Electro-Mechanical System (MEMS) accelerometers are used in diverse applications such as automotive airbag systems, computer disk drives, and home appliances. Newer accelerometers, such as the Analog Devices ADXL202E, are so sensitive that accurate tilt sensing is now possible as well. This example examines the implemention of a level sensor using a PIC16F873 from Microchip Technology, Inc. and an ADXL202E. Testing and calibration will be performed using an Agilent 54642D mixed signal oscilloscope (MSO).

Level sensor concepts

The ADXL202E is a ±2 g dual-axis accelerometer with duty-cycle outputs for the X- and the Y-axes. For the sake of clarity, we will describe only one axis in detail, but keep in mind that both axes are measured and calibrated in precisely the same way.

At 0 g, the output will typically be at 50 percent duty cycle. This is usually referred to as the 0 g bias level. The duty cycle will change approximately 12.5 percent per applied g, depending on which way an axis is tilted. This value is referred to as the sensitivity. The period of the ADXL202E is adjustable and can be set from 0.5 to 10 ms, depending on the application. A typical output and a rough formula for acceleration are shown below.



$$Ax(g) = \frac{\frac{Tx1}{T2} \text{ x } 100-50}{12.5}$$

$$Ay(g) = \frac{\frac{T1y}{T2} \text{ x } 100-50}{12.5}$$

The acceleration can be converted to degrees of tilt as follows:

$$Pitch = \text{arcsin}({}^{Ax}/{}_{1g})$$

$$Roll = \text{arcsin}({}^{Ay}/{}_{1g})$$

Hint 4

Basic calibration

The world isn't a perfect place. It turns out in practice that the 0 g bias level and the sensitivity both vary from their ideal values. Furthermore, these two parameters are VERY temperature dependent, but for now we will assume that our system will stay at room temperature.

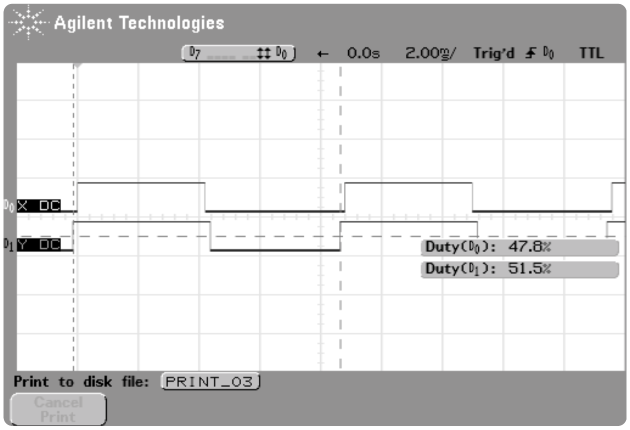
The acceleration equation is better written as:

$$A(g) = \frac{\frac{T1y}{T2} \text{ x } 100-(0g\_Bias(\%))}{Sensitivity(\%)}$$

Placing the accelerometer so that both of its axes are parallel to the surface of the earth, the 0 g bias levels can be recorded. The 0 g bias levels of our test board are shown in Figure 1.

Using gravity as a reference, the device can then be rotated to find the minimum and maximum duty cycles for each axis.

$$Sensitivity(\%) = \frac{MaxDC(\%)-MinDC(\%)}{2}$$



Duty(D0): 47.8%

Duty(D1): 51.5%

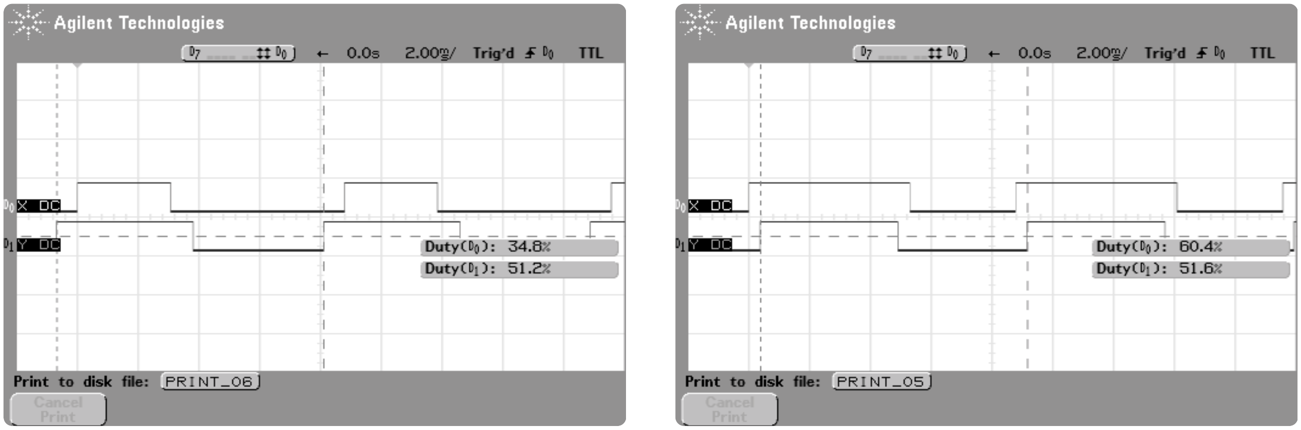
Print to disk file: PRINT_03

X: 0 g bias = 47.8%, Y: 0 g bias = 51.5%

Figure 1. The 0 g bias levels of the test board.

Hint 4

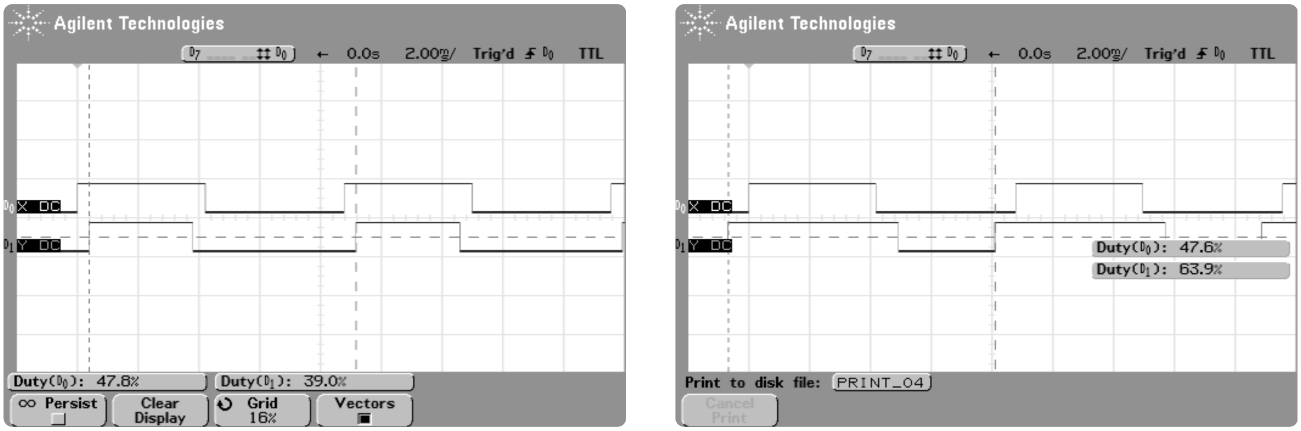
Sensitivity Scope Captures



Duty(D0): 34.8%
Duty(D1): 51.2%

Duty(D0): 60.4%
Duty(D1): 51.6%

Figure 2a. X-axis sensitivity (Min and Max X duty cycles, Y at 0 g)



Duty(D0): 47.8%
Duty(D1): 39.0%

Duty(D0): 47.6%
Duty(D1): 63.9%

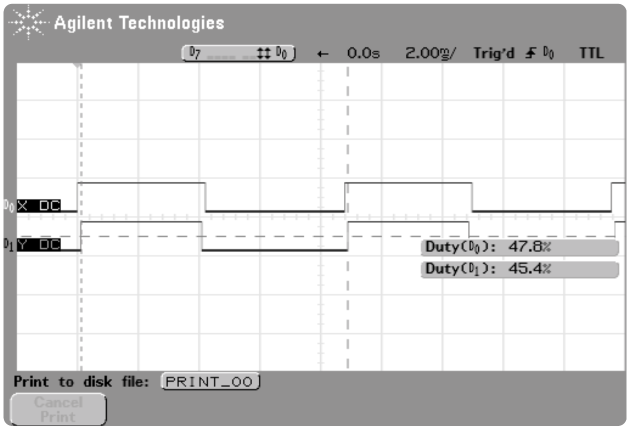
Figure 2b. Y-axis sensitivity (Min and Max duty cycles, X at 0 g)

Measured Values:

	0 g DC	Min DC	Max DC	Sensitivity
X Axis	47.8%	34.8%	60.4%	12.8%
Y Axis	51.5%	39.0%	63.9%	12.45%

Example: 30-degree roll

For a quick example, our test board is "rolled" 30 degrees to the left. The result is shown in Figure 3.



Duty(D0): 47.8%
Duty(D1): 45.4%

Figure 3. Results of "rolling" the test board 30 degrees to the left.

$$Pitch = \arcsin \frac{[47.8\%–47.8\%]}{12.8\%} = 0°$$

$$Roll = \arcsin \frac{[45.4\%–51.5\%]}{12.45\%} = –29.3°$$

Circuit and firmware

Figure 4 is a basic block diagram of the PIC16F873/ADXL202E/MSO setup. The PIC16F873 is an ideal microcontroller for this application due to its internal counters and EEPROM for storing calibration constants. Also included in this diagram is an LM35 temperature sensor from National Semiconductor. It outputs 10 mV/degree C and will be used for a later example.

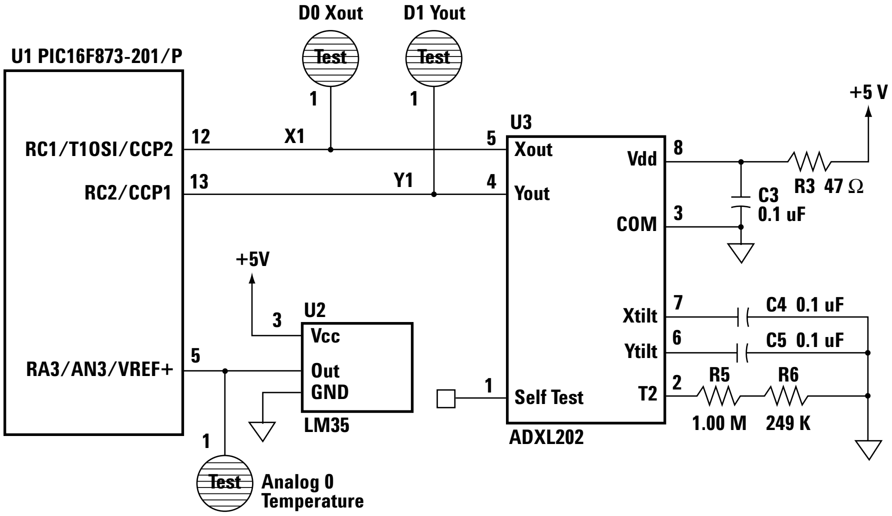


Figure 4. Basic diagram of the test setup.

Hint 4

The job of the PIC microcontroller is to watch for rising and falling edges on the Xout and Yout lines. On the rising edges, the internal counters can be reset to zero and will count until a falling edge is received. The interval between the rising and falling edge will be "T1", while the interval between the rising edges will be the period, or "T2".

Very simple code to monitor the X-axis.

```
float X_Deg(void)
{
    // Measure High pulse width and period of X level
    setup_ccp1(CCP_OFF);                    // Disable CCP1 (Y input)
    setup_ccp2(CCP_CAPTURE_RE);             // Configure CCP2 to capture rise
    CCP2IF=0;                               // Clear CCP2 Flag
    while (!CCP2IF);                        // Wait for rising edge
    set_timer1(0);                          // Clear Timer1
    setup_ccp2(CCP_CAPTURE_FE);             // Capture Fall, Clear Timer1
    CCP2IF=0;                               // Clear CCP2 Flag
    while (!CCP2IF);                        // Wait for falling edge
    pos_pulse = CCP_2;                      // Record pulse width
    setup_ccp2(CCP_CAPTURE_RE);             // Capture rise,
    CCP2IF=0;                               // Clear CCP2 Flag
    while (!CCP2IF);                        // Wait for rising edge
    period = CCP_2;                         // Record period
    setup_ccp2(CCP_OFF);                    // Disable CCP2 (X input)
    duty_cycle = (100 * pos_pulse / period) ;

FB_Angle = (180/3.14159) * asin( (duty_cycle - EEPROM_Read_Float(X_Zero_DC)) /
EEPROM_Read_Float(X_Sensitivity) );

return    FB_Angle;
}
```

Note: 0 g bias and sensitivity points are already stored in EEPROM. FB_Angle and duty_cycle are floating point numbers, pos_pulse and period are 16-bit integers.

The floating point result is returned in FB_Angle and now it is up to the application to determine what to do with it.

Hint 4

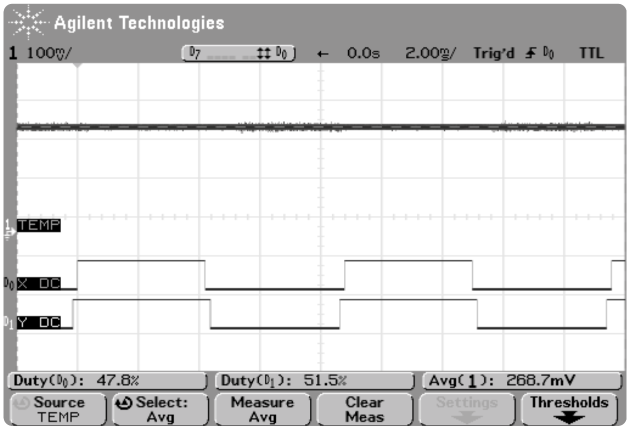
Checking temperature dependence

As briefly mentioned before, the 0 g bias point of the ADXL202E is temperature dependent. Figures 5 through 7 show examples where the only thing that changes is the temperature.

As you can see, the X-axis is more sensitive to temperature than the Y-axis. If your application has to work over a wide temperature range and maintain fine accuracy, temperature compensation will be necessary.

Using deep memory

The Agilent oscilloscope eases testing with its mixed signal capabilities and deep memory. By using the deep memory, it is possible to capture hundreds of output cycles and get a rough idea of the sensor's transient response. In a real-world system, the oscilloscope could also simultaneously monitor several other inputs and outputs to determine system timing.

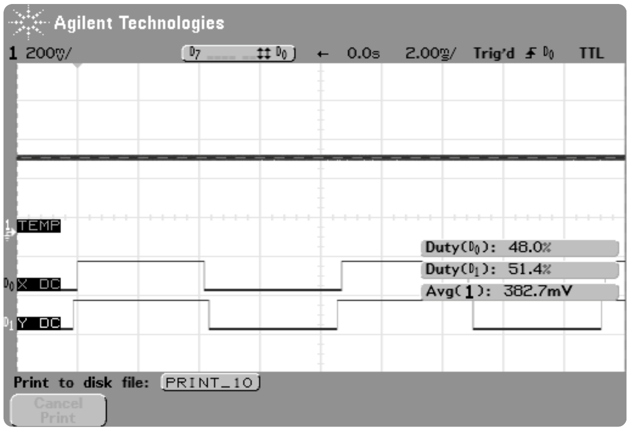


Duty(D0): 47.8%   Duty(D1): 51.5%   Avg(1): 268.7mV

Figure 5. The 0 g bias when the room temperature is 269 mv / 10 mv/C => 27C°

$$Pitch = \arcsin \frac{[47.8\%–47.8\%]}{12.8\%} = 0°$$

$$Roll = \arcsin \frac{[51.5\%–51.5\%]}{12.45\%} = 0°$$



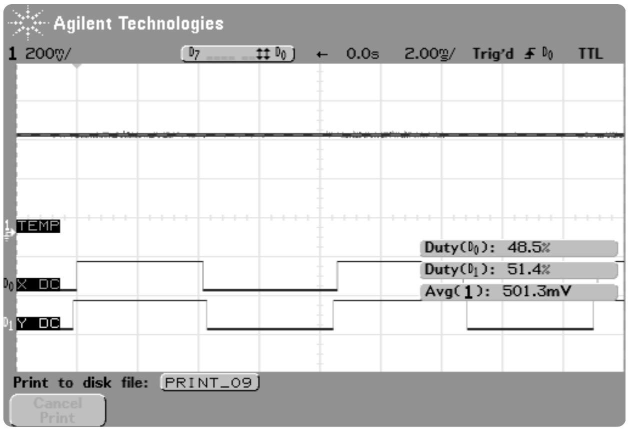
Duty(D0): 48.0%
Duty(D1): 51.4%
Avg(1): 382.7mV

Print to disk file: PRINT_10

Figure 6. The 0 g bias when the room temperature is 383 mv / 10 mv/C => 38C°

$$Pitch = \arcsin \frac{[48.0\%–47.8\%]}{12.8\%} = 0.9°$$

$$Roll = \arcsin \frac{[51.4\%–51.5\%]}{12.45\%} = –0.5°$$



Duty(D0): 48.5%
Duty(D1): 51.4%
Avg(1): 501.3mV

Print to disk file: PRINT_09

Figure 7. The 0 g bias when the room temperature is 501 mv / 10 mv/C => 50C°

$$Pitch = \arcsin \frac{[48.5\%–47.8\%]}{12.8\%} = 3.13°$$

$$Roll = \arcsin \frac{[51.4\%–51.5\%]}{12.45\%} = –0.5°$$

5 Hint

Implementing reliable serial communication with an RC oscillator

By Dave Brobst, Solutions-Cubed

The PIC12FXXX line of microcontrollers from Microchip Technologies, Inc. have an abundance of on-board peripherals, which coupled with their diminutive size and low cost make them a natural fit for small embedded systems. One intriguing feature is the on-board RC oscillator. By using this feature, a system designer does not need to specify a space-hogging and budget-busting clock circuit. However, the RC oscillator is very temperature-dependent, which can preclude its use for clock asynchronous serial communication. This hint shows one way to surmount this obstacle and design a reliable serial communication system.

Asynchronous communication is defined as a serial communication stream without a clock. Essentially, the transmitter and the receiver clock the information using their own locally generated clocks, with no opportunity to synchronize their clocks to each other. With a long communication sequence, small clock differences between the transmitter and the receiver can be magnified, so that the receiver reads the last bits incorrectly. Figure 1 provides a block diagram of a typical serial communication system.

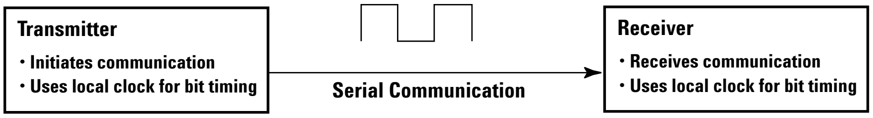

Transmitter
• Initiates communication
• Uses local clock for bit timing
Serial Communication
Receiver
• Receives communication
• Uses local clock for bit timing

Figure 1. Typical serial communication system.

Hint 5

Asynchronous serial data is sent one byte at a time, bit-by-bit. In order to successfully complete the communication, a pre-determined baud rate is used. This is the amount of time that the “on” bit takes. For example, a common baud rate is 9600 bits per second, which means that each bit takes 104.17 µS. In addition, the order of the bits must be pre-determined—either the information is transmitted with the least significant bit first or the most significant bit first. Lastly, an “idle” state of the serial line must be selected—either low or high. Typically, the idle state is high. Most asynchronous communication further defines each byte of data by adding a start and a stop bit. A start bit is defined as a low and a stop bit as a high. Figure 2 shows a byte of serial data transmitted, least significant bit first, at 9600 baud. The byte of data is H’55’.

With a little reflection on Figures 1 and 2, the problem with asynchronous communication can be seen. If both the transmitter and the receiver generate the clocks locally to pace the baud rate, any differences in the clocks are going to be reflected as differences in baud rate for the transmitter and the receiver. At the extreme, if the baud clocks are off enough, the receiver will misread the incoming byte of data.

A simple calculation shows the largest possible difference that the two baud clocks can have before errors begin creeping in. Figure 2 shows that there are 10 bits of data—eight “real” bits and one bit each for start and stop. Assume that the receiver always tries to sample the data at the exact middle of each bit to provide the most “baud rate insensitivity.” Also assume that the clocks stay constant but different from each other for the duration of the transmission. With these assumptions, the calculation shown in Figure 3 gives the maximum clock differences for the transmitter and receiver—or no more than 5 percent—before errors are introduced into the communication.

Here, then, is the crux of the problem. The PIC12FXXX line of microcontrollers specify their internal RC oscillator at ±2 percent tolerance at 2.5 V to 5.5 V and 0°C to 85°C and ±5 percent over the entire voltage and temperature range. So at first blush, the microcontroller’s oscillator, itself, has enough variability to introduce insurmountable communication errors, and this does not even take into account the errors in the clock on the other end of the communication. We will examine one method to easily get around this problem.

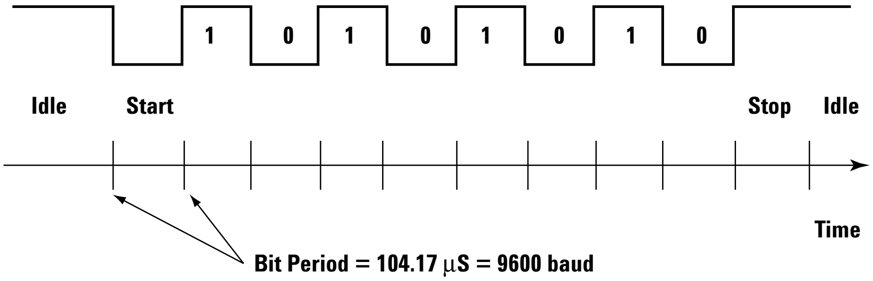


Figure 2. Representative serial byte.

$$\text{Clock Difference Max(\%)} = \frac{\frac{1}{2}\ \text{x Bit Period}}{\text{\# of bits before re-sync}}\ \text{x } 100$$

$$\therefore \text{Clock Difference Max} = 5\%$$

Figure 3. Clock difference calculation.

Hint 5

Luckily in most embedded systems, the designer can define the communication protocol used over the communication channel. A simple method to synchronize the clocks is to pad all communications with a known first byte that the receiver uses to get bit timing information. For example, if every communication from the transmitter to the receiver started with a H’55’, the receiver could use this first byte to determine the bit period that the transmitter was using and from this could synchronize its own baud clock. In order to calculate the correct bit timing, the microcontroller knows that there are five falling edges and five rising edges in the first byte. It can measure the total time for the first byte and divide by the ten bits to get the bit time.

To verify this method of bit time measuring, the Agilent 54622D mixed signal oscilloscope (MSO) proves especially useful. When coupled with the Microchip MPLAB design environment and MPLAB ICE 2000 emulator, the testing process is especially painless. Figure 4 shows a block diagram of the test setup. The test procedure is fairly straightforward. One analog channel of the scope is attached to the serial line to monitor the communication. The other analog channel is attached to the CLKOUT line of the microcontroller to ensure that the clock frequency is at the correct speed. The microcontroller toggles an I/O line every time it reads a serial bit. This is monitored with a digital line on the MSO. To test the extremes, the emulator is used to change the clock frequency. Figure 5 shows how the clock frequency of the microcontroller is changed inside the Microchip design environment.

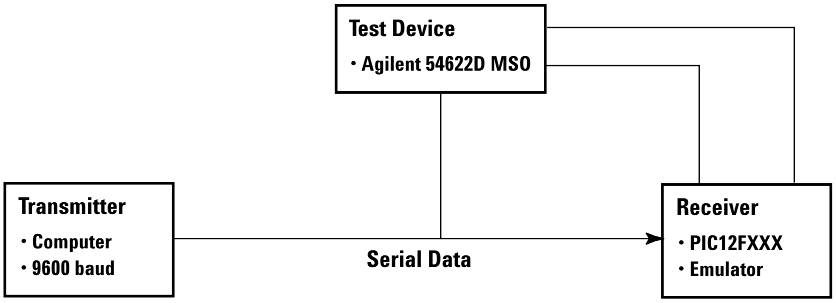
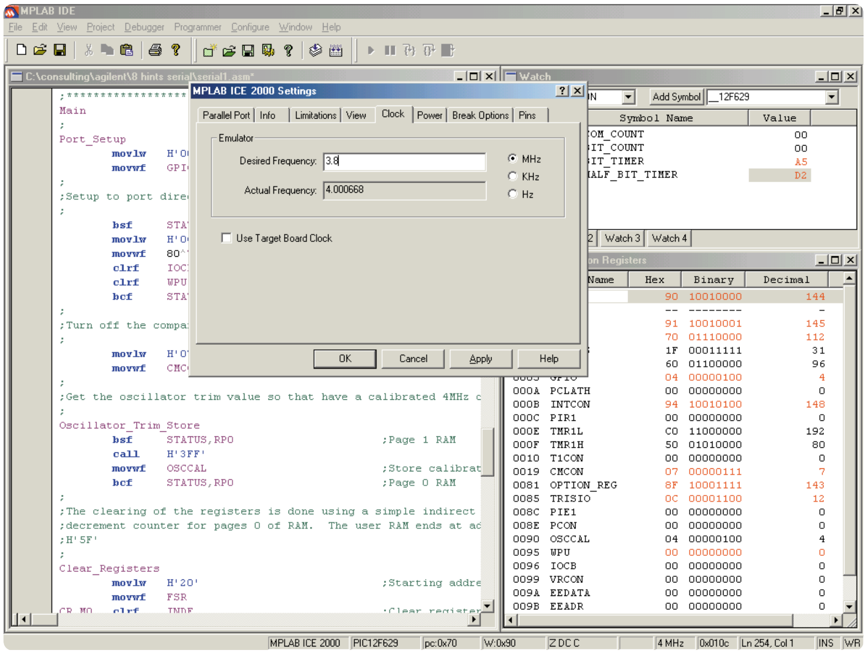


Figure 4. Test connections setup.



```
;******************
Main
;
Port_Setup
        movlw   H'0
        movwf   GPI
;
;Setup to port dire
;
        bsf     STA
        movlw   H'0
        movwf   80^
        clrf    IOC
        clrf    WPU
        bcf     STA
;
;Turn off the compa
;
        movlw   H'0
        movwf   CMC
;
;Get the oscillator trim value so that have a calibrated 4MHz c
;
Oscillator_Trim_Store
        bsf     STATUS,RP0                      ;Page 1 RAM
        call    H'3FF'
        movwf   OSCCAL                          ;Store calibrat
        bcf     STATUS,RP0                      ;Page 0 RAM
;
;The clearing of the registers is done using a simple indirect
;decrement counter for pages 0 of RAM.  The user RAM ends at ac
;H'5F'
;
Clear_Registers
        movlw   H'20'                           ;Starting addre
        movwf   FSR
CR_M0   clrf    INDF                            ;Clear register
```

Emulator

Desired Frequency: 3.8 ● MHz ○ KHz ○ Hz

Actual Frequency: 4.000668

☐ Use Target Board Clock

N Add Symbol __12F629

Symbol Name	Value
OM_COUNT	00
IT_COUNT	00
IT_TIMER	A5
ALF_BIT_TIMER	D2

	Name	Hex	Binary	Decimal
		90	10010000	144
		--	--------	-
		91	10010001	145
		70	01110000	112
		1F	00011111	31
		60	01100000	96
0005	GPIO	04	00000100	4
000A	PCLATH	00	00000000	0
000B	INTCON	94	10010100	148
000C	PIR1	00	00000000	0
000E	TMR1L	C0	11000000	192
000F	TMR1H	50	01010000	80
0010	T1CON	00	00000000	0
0019	CMCON	07	00000111	7
0081	OPTION_REG	8F	10001111	143
0085	TRISIO	0C	00001100	12
008C	PIE1	00	00000000	0
008E	PCON	00	00000000	0
0090	OSCCAL	04	00000100	4
0095	WPU	00	00000000	0
0096	IOCB	00	00000000	0
0099	VRCON	00	00000000	0
009A	EEDATA	00	00000000	0
009B	EEADR	00	00000000	0

MPLAB ICE 2000 | PIC12F629 | pc:0x70 | W:0x90 | Z DC C | 4 MHz | 0x010c | Ln 254, Col 1 | INS | WR

Figure 5. Changing clock frequency on the microcontroller.

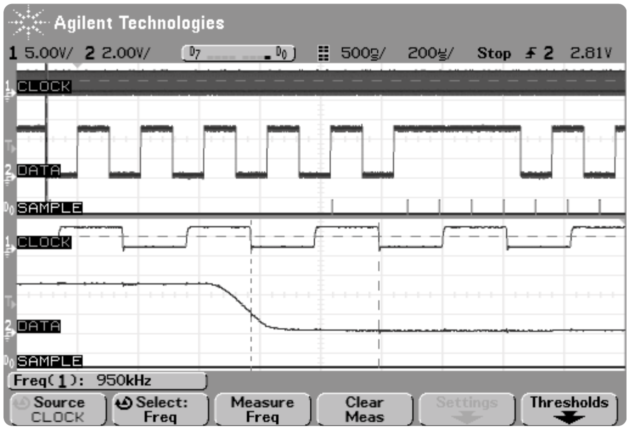
Hint 5

In Figure 6, the operating frequency of the microcontroller is checked. In this case, we are testing a slow 3.8 MHz microcontroller clock. For this internal frequency, the correct CLKOUT frequency should be 950 kHz. By using the Delayed display view we can zoom in on the CLKOUT frequency without re-triggering on the data.

Figure 7 shows how the initial bit time is measured. Note that the D0 line shows when the microcontroller registers the first falling edge and the last rising edge of the synchronization byte. At 9600 baud, this should take 937 µS. Note also that our cursors show the microcontroller measuring 928 µS. This is a better than 1 percent error in measuring the bit timing.

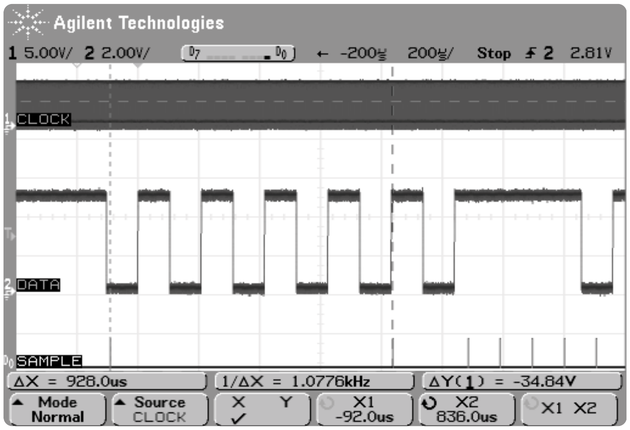
In Figure 8, we can see if the measured bit time is implemented correctly. After the synchronization byte is read, the controller begins to sample on the bit period after the next start bit. The cursors on the D0 line show a 104 µS bit period, which compares favorably with the ideal 9600 baud bit period of 104.17 µS.

Overall, implementing an asynchronous serial system with a PIC12FXXX using the internal RC oscillator can be accomplished with some thoughtful planning. When verifying the system, a multi-channel oscilloscope, such as the Agilent 54622D, with a deep memory and user friendly features like cursors, delayed viewing, and quick measurements is an invaluable debugging tool.



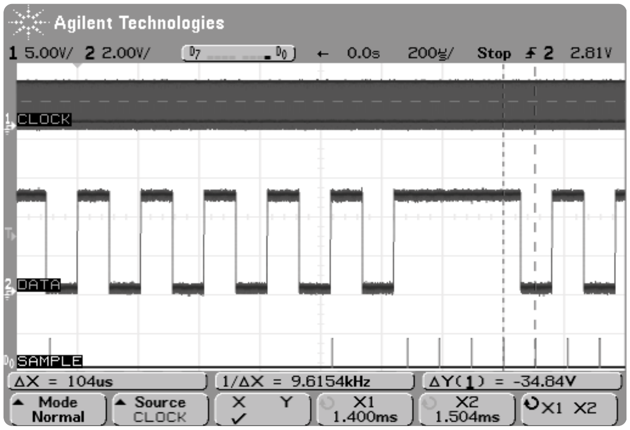
Freq(1): 950kHz

Figure 6. Verifying the clock frequency.



ΔX = 928.0us 1/ΔX = 1.0776kHz ΔY(1) = -34.84V

Figure 7. Checking the synchronization word.



ΔX = 104us 1/ΔX = 9.6154kHz ΔY(1) = -34.84V

Figure 8. Showing the controller is measuring the bit period correctly.

Mixed Analog and Digital Signal Oscilloscopes

Seamless integration of scope and timing channels

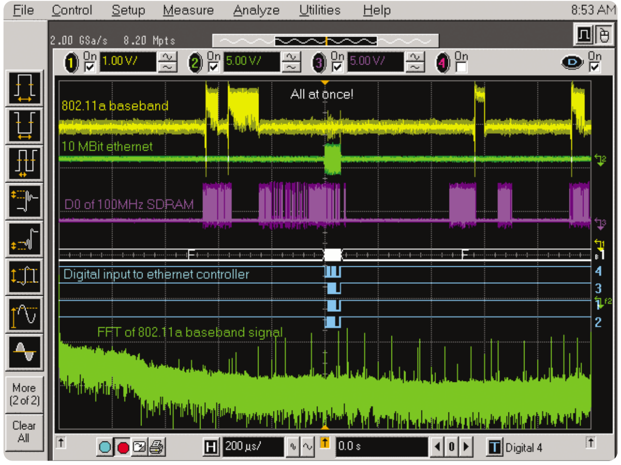


Figure 1. MegaZoom allows you to capture a full cycle of a device's operation while maintaining the ability to resolve fine details.

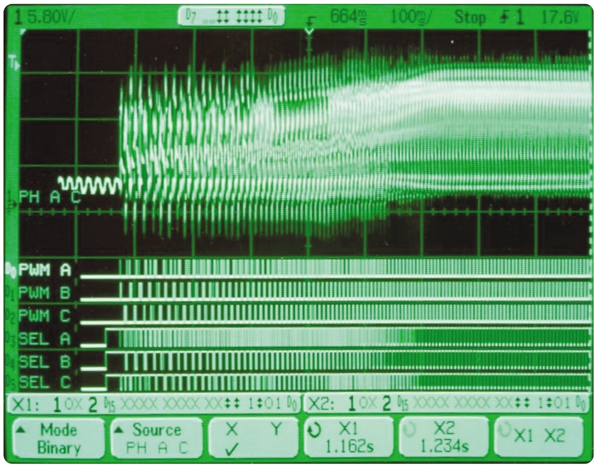
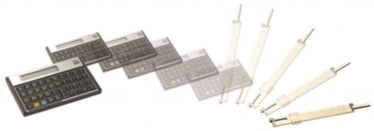


Figure 2. A high-resolution, analog-like display with 32-shades of gray helps you easily identify waveform irregularities (Agilent 54600 Series only).

- Choose from bandwidths ranging from 60 MHz to 1 GHz
- View 18 to 20 time-aligned scope and timing channels simultaneously
- Troubleshoot mixed analog and digital systems with responsive MegaZoom deep memory
- Reveal subtle details that the typical scope won't show with a patented high-resolution display
- Take advantage of standard serial triggering including I^{2}C, SPI, CAN frame, and USB frame



Using a scope without logic channels instead of an Agilent MSO is like using a slide rule when you could have a calculator!

More channels, more memory, more triggering

With the increasing digital content in today's designs, it is often difficult to capture enough channels simultaneously with a traditional 2- or 4-channel scope. To further complicate matters, the analog and digital signals are often operating at drastically different speeds.

Now you can capture, display, and analyze a variety of analog and digital signals in one acquisition on one instrument screen, helping you narrow in more quickly on tough design problems. With mixed signal oscilloscopes (MSOs), a 16-channel timing analyzer is seamlessly integrated into a full-featured scope. It's now easy to measure a combination of signal types and speeds all at once, including slow analog, fast digital, or baseband RF.

MegaZoom memory

MegaZoom memory technology is fast *and* deep, so you can capture a full cycle of your device's operation with the resolution needed to view critical intervals of the highest speed signals. MegaZoom is available at all times and does not require a special operating mode.

Mixed Analog and Digital Signal Oscilloscopes

Seamless integration of scope and timing channels

Figure 3. Agilent's family of mixed signal oscilloscopes spans the frequency range from 60 MHz to 1 GHz, with models to meet your needs and budget.

	54832D Infiniium	54831D Infiniium	54830D Infiniium	54642D Portable	54641D Portable	54622D Portable	54621D Portable
Bandwidth	1 GHz	600 MHz	600 MHz	500 MHz	350 MHz	100 MHz	60 MHz
Channels Scope+Timing	4+16	4+16	2+16	2+16	2+16	2+16	2+16
Sampling	4 GSa/s	4 GSa/s	4 GSa/s	2 GSa/s	2 GSa/s	200 MSa/s	200 MSa/s
Maximum Memory	Up to 16 M	Up to 16 M	Up to 16 M	8 M	8 M	4 M	4 M
MegaZoom	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Special Features	Infiniium's Windows®-based interface gives you unparalleled ease of use for both scope and logic channels. Powerful triggering, measurements, and math functions make this the scope of choice for embedded applications. LAN and Web connectivity let you share information easily and control the scope remotely.			High-definition display with 32 shades of gray give a clear analog-like view of critical waveforms. Familiar knob-based controls for both scope and logic channels helps you put these scopes to work right out of the box. Serial triggering simplifies tracking down CAN, I^{2}C, SPI, USB and other serial-bus problems. IntuiLink software simplifies PC connectivity.			
Description	For high-performance 32-bit embedded applications with high-speed logic	For 32-bit applications with logic; ~2 ns edge speeds and 4 scope channels	For DSP-based systems requiring extended analysis and deep memory	For higher-speed embedded and DSP applications	For medium-speed embedded applications up to 50 MHz	For 8- and 16-bit microcontroller applications	The lowest-cost MSO; ideal for education and industrial applications
Notes	Standard memory is 4 M, options extend up to 16 M.			IntuiLink software is provided at no cost. Scopes have built-in parallel and RS-232 I/O. GPIB is an extra-cost option.			

Table 1. Mixed signal selection guide.

For more information about Agilent MSOs, go to **www.agilent.com/find/mso**

Related Literature

Publication Title	Publication Type	Publication Number
Agilent Technologies 54600 Series Oscilloscopes	Data Sheet	5968-8152EN
Agilent Technologies Infiniium 54800 Series Oscilloscopes	Brochure	5988-3788EN
Infiniium 54850 Series Oscilloscopes *InfiniiMax 1130 Series Probes*	Data Sheet	5988-7976EN
Infiniium 86100B DCA: Modular Platform for Fast, Accurate Waveform Testing up to 40 Gb/s	Brochure	5988-5235EN

The following literature provides useful information on using oscilloscopes for specific applications.

Optimizing Oscilloscope Measurement Accuracy on High-Performance Systems with Agilent Active Probes	Application Note 1385	5988-5021EN
Debugging a PCI Bus with a Mixed-Signal Oscilloscope	Application Note 1417	5988-7745EN
Mixed Analog and Digital Signal Debug and Analysis Using a Mixed-Signal Oscilloscope Wireless LAN Example Application	Application Note 1418	5988-7746EN
Finding Hidden Problems Using Agilent's Deep-Memory Oscilloscope: How IBM Solved a Mystery	Customer Success Story	5988-5655EN
The Truth About the Fidelity of High-Bandwidth Voltage Probes	Application Note 1404	5988-6515EN
Verifying Bluetooth™ Baseband Signals Using Mixed-Signal Oscilloscopes	Application Note 1333-3	5988-2181EN
Spectral Analysis Using a Deep-Memory Oscilloscope Fast Fourier Transform (FFT) For Use with Infiniium 54830B Series Deep-Memory Oscilloscopes	Application Note 1383-1	5988-4368EN
54622D Agilent I^2C Debugging	Application Note1351	5980-0796EUS
Debugging Digital Cameras: Detecting Redundant Pixels	Application Note	5988-3358ENUS
54622D Agilent MSO and CEBus PL Communications Testing	Application Note 1352	5980-0794EUS
Debugging Serial Bus Systems with a Mixed-Signal Oscilloscope	Application Note 1395	5988-5997EN
54622D Agilent Debugging Modern Power Electronics: Seeing the Whole Picture	Application Note 1350	5980-0797EUS
Agilent Infiniium Oscilloscopes Performance Guide Using 89601A Vector Signal Analyzer Software	Product Note	5988-4096EN

Agilent Technologies' Test and Measurement Support, Services, and Assistance

Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise

Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage

Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

Agilent T&M Software and Connectivity

Agilent's Test and Measurement software and connectivity products, solutions and developer network allows you to take time out of connecting your instruments to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit **www.agilent.com/find/connectivity** for more information.

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) 800 452 4844

Canada:
(tel) 877 894 4414
(fax) 905 282 6495

China:
(tel) 800 810 0189
(fax) 800 820 2816

Europe:
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

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

Korea:
(tel) (82 2) 2004 5004
(fax) (82 2) 2004 5115

Latin America:
(tel) (305) 269 7500
(fax) (305) 269 7599

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6836 0252
Email: tm_asia@agilent.com

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

Printed in USA May 9, 2003
5988-9489EN