

Edited by Bill Travis and Anne Watson Swager

Square-root function improves thermostat

W Stephen Woodward, University of North Carolina, Chapel Hill, NC

PERHAPS THE MOST elementary rule of control-loop design theory is that feedback-loop performance is fundamentally linked to the careful choice—and stability—of loop gain. Insufficient loop gain leads to poor setpoint accuracy. Too much gain can induce feedback instabilities, such as overshoot, ringing, and, ultimately, oscillation.

Therefore, the greater the accuracy you expect from a control system, the more critical maintaining near-optimal loop gain becomes. Precision temperature-control loops are no exception. Given the aforementioned truisms, it's surprising that the following rule of designing high-precision thermostats receives so little notice: The thermal output (which is power, the primary feedback parameter) from a resistive heater is proportional to the current squared. In **Figure 1**, Curve A illustrates this elementary relationship. Therefore, the overall thermostat loop gain is not constant but is instead proportional to heater input current. It consequently varies wildly in response to changes in ambient temperature and other factors that impact heat demand. The

result is that it becomes more difficult to choose suitable loop parameters. The circuit in **Figure 2** (pg 114) remedies these difficulties by inserting an analog square-root circuit ahead of the heater-drive circuit.

The circuit stabilizes the temperature of liquid-nitrogen-cooled solid-state in-

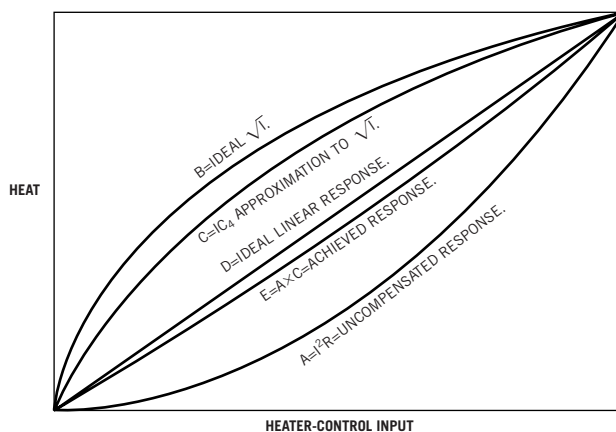
through IC_{4C} combine the current, I₁, from Q₁ with the reference current (I_{REF}) to produce a logarithmic control voltage proportional to $\log(I_1 \cdot I_{REF})/2 = \log(\sqrt{I_1 \cdot I_{REF}})$. The inherent matching of transistor parameters in the IC₄ monolithic array results in an IC_{4E} collector current of approximately $\sqrt{I_1 \cdot I_{REF}}$. The

IC_{3B}-Q₂ heater-driver circuit subsequently amplifies IC_{4E}'s output current by a factor of 8450 and applies the amplified current to the laser-cryostat heater.

Figure 1 shows five relevant curves. A is the uncompensated I²R heater transfer function. B is the ideal square-root function. C is the square-root approximation from the IC₄ array, which you calculate assuming transistor betas of approximately 100. D is the product of A and B and represents the ideal compensated (linear) loop-gain linearization with constant loop

gain. E is the product of A and C and is the achieved loop-gain linearization. The net result is a linear relationship between the control circuit and heater outputs and a consequent optimization of the cryostat's steady-state and dynamic stabilities over a range of ambient-heat loading. Without IC₄, a gain setting adequate for setpoint stability at low heater powers is likely to be excessive and produce overshoot or oscillation at high heater powers. (DI #2417).

Figure 1



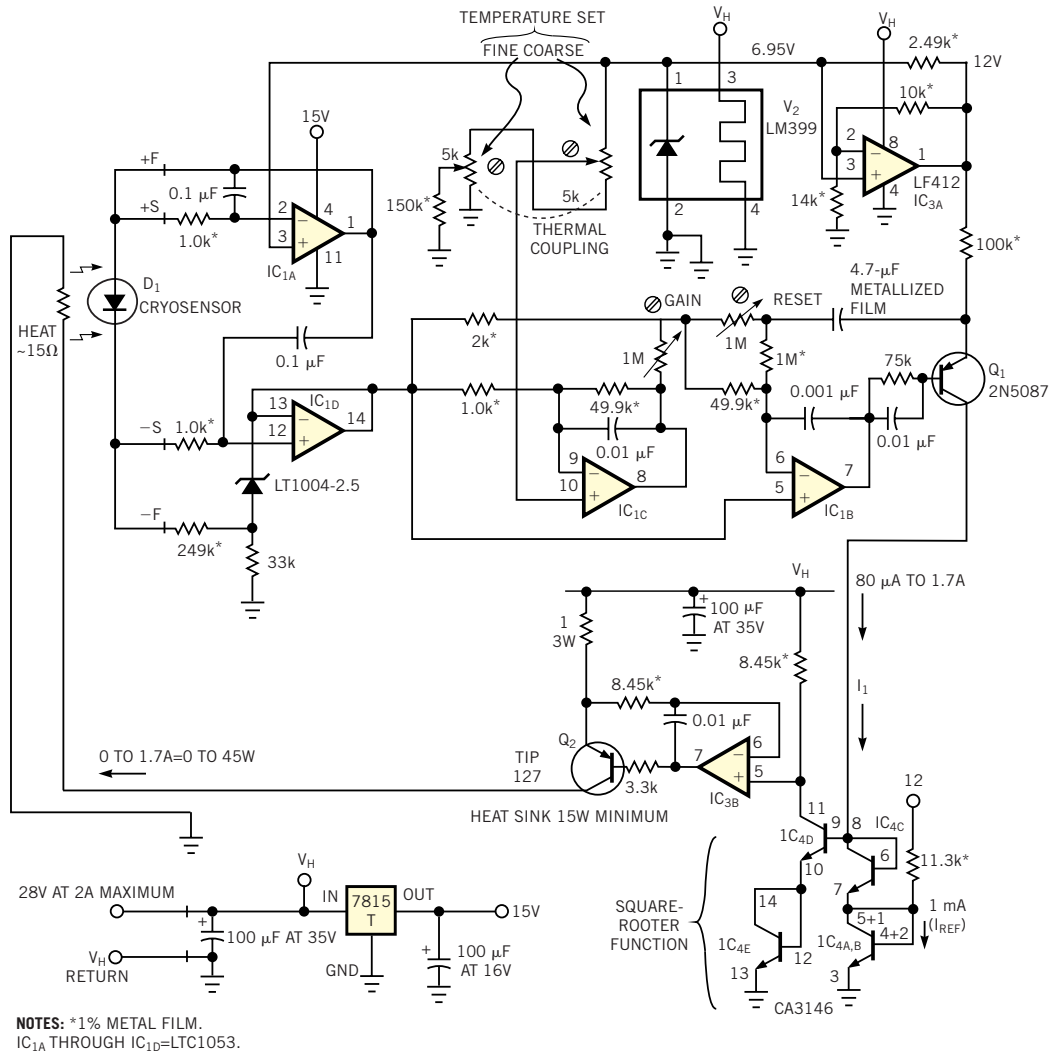
Multiplying the uncompensated square-law response (A) by the square-root current response (C) from IC₄ yields a linear response of heat versus heater-control output.

frared lasers in an airborne spectrometer. The cryosensor diode, D₁ (2 mV/K), senses laser temperature and drives the PI (proportional-integral) control circuit comprising error amplifier IC_{1C} and error integrator IC_{1B}. Q₁ converts the resulting feedback correction voltage to a current-mode signal and applies the signal to the LM3146 transistor array, IC₄. The array generates the square-root function. Analog aficionados will be quick to point out that using IC₄ is not the most accurate way to approximate a square-root curve. However, this method is adequate for making the feedback linear and stabilizing loop-gain stabilizing the loop gain. In operation, array transistors IC_{4A}

Square-root function improves thermostat	113
µC detects transmission rate of RS-232 interface	114
Current sensor measures 0 to 500A	118
Safely swap SCSI disk drives	120
SSB modulator covers HF band	122
Switched-capacitor IC forms notch filter	124

TO VOTE FOR THIS DESIGN,
CIRCLE No. 365

Figure 2



The logarithmic response of transistors IC_{4A} through IC_{4C} results in a square-root function for the heater-control voltage.

µC detects transmission rate of RS-232 interface

by Thomas Schmidt, Microchip Technology, Chandler, AZ

RS-232 IS THE most common serial interface in the PC world. Most RS-232 interfaces communicate with the receiver at a fixed transmission rate, such as 9600 baud. But what happens if the transmitter operates with different transmission rates? Different transmission rates require the receiver to detect the rate and adjust the software to the new communication speed. The follow-

ing description of how a receiver detects the transmission rate of an RS-232 interface does not describe the implementation of a receive-and-transmit routine. Instead, it describes a system consisting of a transmitter and a receiver. The transmitter (for example, a PC) transmits a character to the receiver. The receiver, a low-cost µC, detects the transmission rate and adjusts its software according to

the new rate. The theory of implementation is simple.

The transmitter sends a calibration value to the receiver. The receiver measures the time to receive the bits of the calibration value. Based on this measurement, the receiver calculates the transmission time of 1 bit. The method uses this time for the baud-rate generator. The trick is to measure the time of the

LISTING 1—RS-232 TRANSMISSION-RATE-DETECTION ROUTINE

```

Autobaud      clr  AUTOBAUD_LOW  ; reset register
              clr  AUTOBAUD_HIGH ; reset register
              clr  AUTOHALF_LOW  ; reset register
              clr  AUTOHALF_HIGH ; reset register
              clr  AUTOB_STATUS  ; reset autobaud status
              ; register

TestStartBit  btfsc PORTA, RX ; check for start-bit
              goto TestStartBit ; Start-bit not found

TestBitHigh   btfsc PORTA, RX ; Test for end of bit
              ; stream
              goto Calculate   ; End of bit stream, now
              ; calculate
              ; bit time for one bit

              incfsz AUTOBAUD_LOW, f ; increment Autobaud
              ; low register
              goto TestBitHigh ; test for high bit
              incfsz AUTOBAUD_HIGH, f ; increment high byte
              ; of autobaud
              ; register
              goto TestBitHigh ; test for end of bit stream
              goto Signal2Slow  ; High byte got an overflow.
              ; Transmitted
              ; signal is too slow for clock
              ; speed of the uC

              ; divide by measure time by 6 (6 one's where transmitted)
Calculate     movlw 0x03 ; initialize count register
              movwf COUNTER ; Counter for number for rotates
              ; = 3

Divide bcf STATUS, C ; clear carry bit
          rrf AUTOBAUD_HIGH, f ; rotate autobaud high register
          rrf AUTOBAUD_LOW, f ; rotate autobaud low register
          decfsz COUNTER, f ; decrement counter
          goto Divide ; divide

              ; Calculate half the bit time
CalcHalfBit   bcf STATUS, C ; clear carry bit
          rrf AUTOBAUD_HIGH, w ; rotate autobaud high register
          movwf AUTOHALF_HIGH ; copy result into AUTOHALF_HIGH
          ; register
          rrf AUTOBAUD_LOW, w ; rotate autobaud high register
          movwf AUTOHALF_LOW ; copy result into AUTOHALF_LOW
          ; register

AdjustLowByte movlw 0x3 ; 18-19 instruction cycles
              ; overhead from
              ; transmit and receive routine.
              subwf AUTOBAUD_LOW, f ; Adjust low byte from Autobaud
              ; counter

              btfss STATUS, C ; Is result negative? (equal=0
              ; will be checked
              ; at ErrorCheck), C=0 result is
              ; negative
              goto Signal2Fast ; Signal is too fast for receive
              ; and transmit routine
              movlw 0x02 ; subtract 2 from low byte of
              ; half the bit time
              subwf AUTOHALF_LOW, f ; subtract from low byte of half ; the bit
              time
              btfss STATUS, C ; Is result negative? (equal=0
              ; will be checked
              ; at ErrorCheck), C=0 result is
              ; negative
              goto Signal2Fast ; Signal is too fast for receive
              ; and transmit routine

              ; check if AUTOBAUD_HIGH and AUTOBAUD_LOW are zero. This
              ; means the transmission time for one byte is too high
ErrorCheck    movf AUTOBAUD_HIGH, w ; copy high byte of autobaud
              ; counter register into
              ; w-register
              xorwf AUTOBAUD_LOW, w ; AUTOBAUD_HIGH = AUTOBAUD_LOW?
              btfss STATUS, Z ; is result zero?
              goto ErrorCheckHalf ; Result is not zero, therefore
              ; finish autobaud routine
              goto Signal2Fast ; Signal is too fast for routine
ErrorCheckHalf movf AUTOHALF_HIGH, w ; copy high byte of
              ; autobaud
              ; counter register into
              ; w-register
              xorwf AUTOHALF_LOW, w ; AUTOBAUD_HIGH =
              ; AUTOBAUD_LOW?
              btfss STATUS, Z ; is result zero?
              goto EndAutoBaud ; Result is not zero,
              ; therefore finish autobaud
              ; routine

              ; Error: delay for half the bit time is zero, therefore a
              ; delay cannot be generated with the delay routines.
              ; incoming signal is too fast for clock speed.
Signal2Fast   bsf AUTOB_STATUS, SIGNAL_FAST ; set error flag
              retlw 0x00 ; return to OS

Signal2Slow   bsf AUTOB_STATUS, SIGNAL_SLOW ; set error
              ; flag

EndAutoBaud   retlw 0x00 ; Return to operating system
    
```

incoming bit stream and calculate the average time to receive 1 bit. This implementation of an autobaud routine assumes that the receiver knows the bit sequence of the calibration value and that the receiver knows when to calibrate. The technique uses a PIC16C54B μ C. The μ C connects to a PC via a MAX232 chip. The PC sends the calibration character to the μ C. We chose the ASCII value of “?” because of the bit sequence (00111111). The autobaud routine measures the time to receive the ones in the bit stream and then divides the time by six. The result is the time the routine takes to receive or transmit 1 bit.

Because the PIC16C54B has no hardware USART, a software routine measures the timing of the bit sequence. **Listing 1** gives the source code of the autobaud routine. The calibration char-

acter contains one start bit, one stop bit, and no parity bit. For time measurement, the technique uses a 16-bit counter, which provides a range of transmission speeds. In the first part of the routine, the software initializes the counter and an autobaud-status register, AUTOB_STATUS. The register stores information about whether the incoming signal is too slow or too fast for the autobaud routine. You can use this information to check whether the calibration process is successful. After the initialization, the autobaud routine looks for the start bit, which is a logic-one-to-zero transition. After detecting the start bit, the autobaud routine looks for the reverse transition. Following detection of this transition, the routine starts measuring the time, using the 16-bit software counter. The software increments the low byte of the 16-bit

counter until the counter overflows.

When an overflow occurs, the high byte of the 16-bit counter increments by one. This process continues until either a change from logic one to zero occurs or the high byte of the counter overflows. In either of these cases, the routine sets a flag in AUTOB_STATUS to indicate that the incoming signal is fast or slow. Otherwise, the software calculates the transmission time of 1 bit. This time generates the baud rate for the transmit or receive routine. These routines need the transmission time of 1 bit either for generating a delay for bit sampling or for bit transmission. The software calculates the transmission time for 1 bit by dividing the measured time by the number of transmitted ones in the calibration value. In the case of the calibration value “?”, it’s necessary to divide the measured time by

six. Dividing by six entails shifting the 16-bit counter/register three times to the right while drawing zeros from the left. After the division, the routine divides the bit time by two, to calculate the transmission of half a bit. This time figure serves in the receive routine to place the bit sampling in the middle of a bit. The division by two entails a simple shift of the 16-bit counter by one position to the left. The program stores the results of this operation in two registers: AUTOHALF_LOW and AUTOHALF_HIGH.

Once the program completes this calculation, it's necessary to adjust the transmission time of 1.5 bits to the software overhead. This adjustment involves subtracting the number of instruction cycles it takes to execute either the transmit or the receive routine. After the subtraction, the software verifies whether the result is smaller than zero. If so, the incoming signal is too fast, and the routine sets an error flag in the AUTOB_STATUS register. After the adjustment, the software verifies whether the incoming signal is too

fast by verifying that the value of the 16-bit counter is zero. If the incoming signal is not too fast, the autobaud routine returns to the operating system. **Listing 1** is available for downloading from EDN's Web site, www.ednmag.com. Click on "Search Databases" and then enter the Software Center to download the file for Design Idea 2418. (DI #2418).

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 366

Current sensor measures 0 to 500A

Jose Carrasco, Universidad de Valencia, Department Electronica, Valencia, Spain

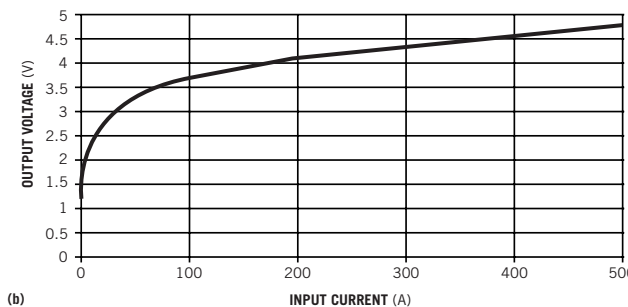
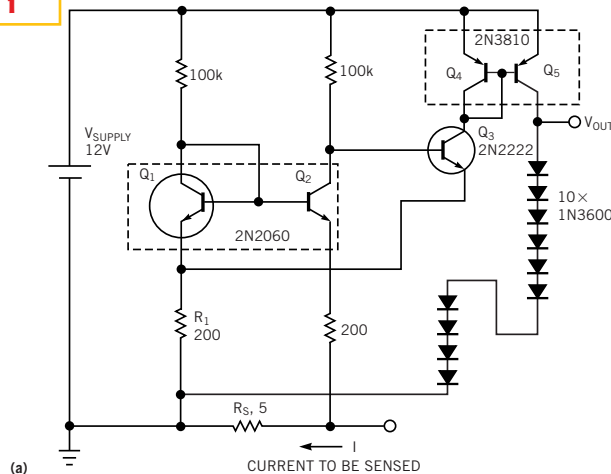
THE SIMPLE CIRCUIT in **Figure 1a** can sense both low and high current levels without low sensitivities or loss of accuracy either at the low or the high end of the scale. The circuit is useful for discerning either low or high currents in noisy environments.

The circuit comprises a current mirror formed by complementary pair Q_1 and Q_2 and the feedback provided by Q_3 . When a current I flows through R_S , the voltage at the emitter of Q_2 increases. The voltage at the base of Q_3 then increases, which increases the current, I_{EQ3} , through Q_3 's emitter. This process continues until the circuit restores its equilibrium by positioning Q_2 at the same operating point as Q_1 , which is working as a diode. Consequently, the following relationship holds:

$$I \cdot R_S = I_{EQ3} \cdot R_1$$

Therefore, Q_3 's emitter delivers a current propor-

Figure 1



A simple current sensor (a) can measure currents of 0 to 500A with a logarithmic output (b).

tional to the current through R_S . Further, because Q_4 also works as a diode, Q_5 has to work at the same operating point as Q_4 so that the current that the collector of Q_5 delivers is also proportional to I . The collection of series diodes provides the current-to-voltage converter in logarithmic scale. **Figure 1b** shows the dc transfer function, which is the output voltage, V_{OUT} , versus the current through R_S .

It is important to use complementary pairs for Q_1 - Q_2 and Q_4 - Q_5 because the operating point of each transistor in the pair should be the same, even if temperature varies within its unions. The circuit uses low-power passive components except for the 5-m Ω resistor R_S , which is manufactured with calibrated wire of well-known ohms/meter characteristics.

(DI #2405)

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 367

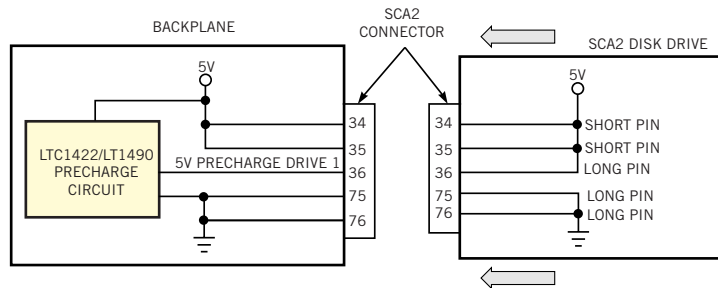
Safely swap SCSI disk drives

By Ajmal Godil, Linear Technology Corp, Milpitas, CA

WHEN YOU INSERT an SCA2 (single-connector attach) drive into a live backplane, the power supply's bypass capacitors in the drive can draw huge transient currents from the backplane's power bus as they charge. The transient currents may cause permanent damage to the connector pins and glitches in the system supply, thereby causing other boards in the system to reset. A viable option for solving this problem is to slowly increase the supply voltage as the drive mates with the backplane. According to the SFF8046 specification, an SCA2-drive connector should have a 5V precharge pin, allowing precharging to occur before the voltage pins make contact (Reference 1, Figure 1). A simple method for providing the precharge voltage is to use a power resistor from the 5V line to the precharge pin. However, after the precharge cycle completes and before the 5V pin mates with the precharge pin, some drives draw as much as 1A of load current. The voltage drop across the power resistor prevents a full precharge and results in a glitch in the 5V backplane supply when the power pin finally mates.

Figure 2 shows a solution to the 5V-precharge problem, using an LTC1422 hot-swap controller and using an LT1490 as a comparator. The output capacitor, C_1 , charges to $4.3V (5V - V_{DIODE})$ through

Figure 1



The SFF8046 specification for the SCA-2 connector in SCSI disk drives calls for a 5V precharge pin, allowing precharging of capacitors to take place before the voltage pins make contact.

R_2 and D_1 . IC_{2A} 's output, Pin 1, is low, and Pin 7 is high. When you insert the drive and it mates with the 5V precharge-drive 1 pin, the drive pulls the voltage on C_1 to ground. This action forces IC_{2A} 's Pin 1 to switch high, thereby turning on the LTC1422. The load current then starts ramping up. At approximately 1A, the voltage drop across R_1 is sufficient to trip IC_{2B} 's output, Pin 2, low, thus keeping the LTC1422's On pin high. Q_1 's source voltage rises to 5V, and the cathode of D_3 (5V precharge-drive 1 pin) rises to 4.7V. Reducing or increasing the value of C_2 can shorten or lengthen the output-voltage turn-on time, respectively. When you fully insert the drive, the 5V line mates with the 5V precharge-drive 1 pin and reverse biases D_3 , allowing the output voltage to

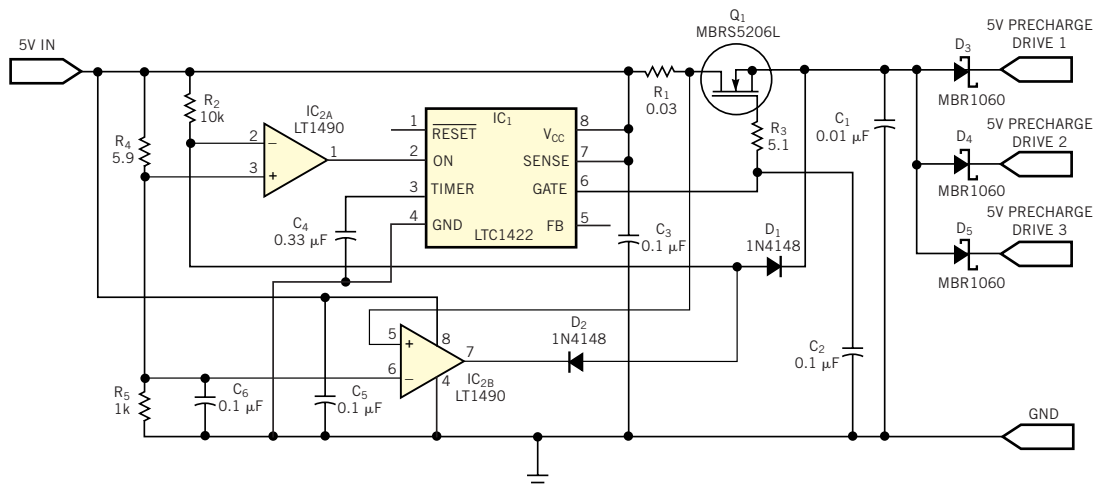
jump from 4.7 to 5V. The load current through Q_1 drops to zero, thereby toggling IC_{2B} 's Pin 7 high and Pin 1 to a low state. This action shuts off the gate driver in the LTC1422, and the chip powers down. The output on C_1 falls to 4.3V, and, as soon as Drive 2 mates with the 5V precharge-drive 2 pin, the LTC1422 powers up, providing a controlled ramp-up output voltage. The process repeats for drives 3, 4, and so on. (DI #2419).

REFERENCE

1. SFF Committee, SFF8046 specification for 80-pin SCA-2 connector for SCSI disk drives, Revision 2.7, Oct 3, 1996.

TO VOTE FOR THIS DESIGN,
CIRCLE No. 368

Figure 2



A hot-swap-controller IC and a comparator provide a controlled ramp-up of the MOSFET's gate voltage and, consequently, the output voltage.

Switched-capacitor IC forms notch filter

Luca Vassalli, Maxim Integrated Products, Sunnyvale, CA

YOU CAN USE A switched-capacitor lowpass filter (LPF) to implement an inexpensive notch

filter (Figure 1a). The internal architecture of the IC (Figure 1b) includes summing nodes similar to those nodes that analog-signal-processing stages use for feedback-error generation. The IC lowpass-filters the quantity $V_{IN} - V_{COM}$ and adds V_{OS} at the output. In other words,

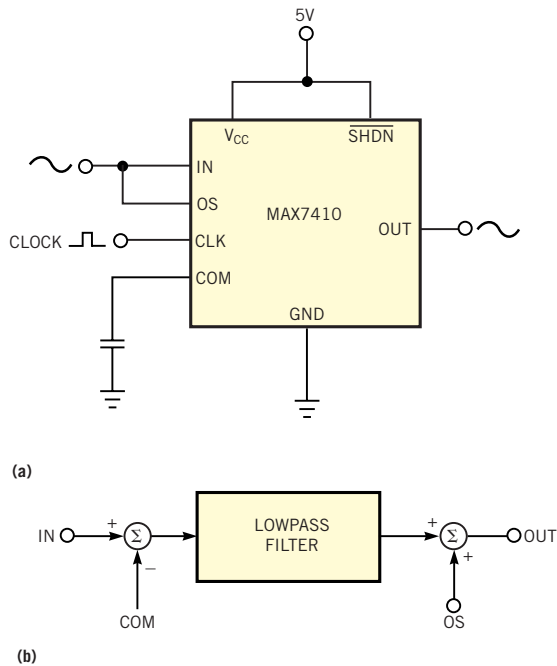
$$V_{OUT} = (V_{IN} - V_{COM})_{LPF} + V_{OS}$$

where V_{COM} typically equals $V_{DD}/2$. Thus, the IC adds common-mode voltage (COM) at the input, and an internal resistor divider biases this voltage at mid-supply. For applications that require offset adjustment or dc-level-shifting, the IC adds an external bias voltage (OS) at the output.

To obtain a notch response, you simply apply the input signal to both the OS and IN pins of the IC, so that these two signals sum at the output (Figure 1a). Frequencies at OS are limited to about 100 kHz. At low frequencies, the output amplitude equals $|V_{IN} + V_{OS}| = 2|V_{IN}|$. At the frequency for which the lowpass filter's phase response changes by 180° , the two signals sum to near zero, creating the notch frequency. The circuit can easily produce a notch at frequencies of 1 Hz to 10 kHz. At higher frequencies, only the OS signal passes through to the output.

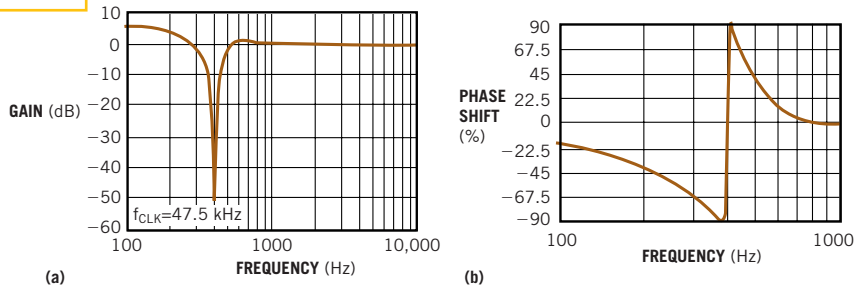
Figure 2a shows the filter response for a 400-Hz notch and a clock frequency of 47.5 kHz. The Q is 1.7, and the center-frequency attenuation is approximately -50 dB. Ripple in the passband limits the notch depth so that the IC's flat passband in the lowpass configuration suits this application. The 180° phase shift occurs at $0.85 f_c$

Figure 1



Simple connections produce a notch filter response (a). The IC's internal summing nodes add a $V_{DD}/2$ common-mode voltage and an externally applied offset voltage (b).

Figure 2



Applying a 47.5-kHz clock to the circuit produces a 400 Hz notch filter with a notch depth of 50 dB and a Q of 1.7 (a). A 180° phase shift occurs at $0.85 f_c$ (b).

, giving the response a smooth -6 -dB transition between the prenotch and postnotch frequencies (Figure 2b). The usable input bandwidth is $f_{clk} - f_{clk}/100$, thanks to a 100-to-1 ratio between the f_c

and clock frequencies. As with all sampling systems, you must take care to avoid input-signal aliasing. (DI #2416)

TO VOTE FOR THIS DESIGN,
CIRCLE NO. 370

Design Idea Entry Blank

Entry blank must accompany all entries. \$100 Cash Award for all published Design Ideas. An additional \$100 Cash Award for the winning design of each issue, determined by vote of readers. Additional \$1500 Cash Award for annual Grand Prize Design, selected among biweekly winners by vote of editors.

To: Design Ideas Editor, EDN Magazine
275 Washington St, Newton, MA 02158

I hereby submit my Design Ideas entry.

Name _____

Title _____

Phone _____

E-mail _____ Fax _____

Company _____

Address _____

Country _____ ZIP _____

Design Idea Title _____

Social Security Number _____
(US authors only)

Entry blank must accompany all entries. (A separate entry blank for each author must accompany every entry.) Design entered must be submitted exclusively to *EDN*, must not be patented, and must have no patent pending. Design must be original with author(s), must not have been previously published (limited-distribution house organs excepted), and must have been constructed and tested. Fully annotate all circuit diagrams. Please submit software listings and all other computer-readable documentation on a IBM PC disk in plain ASCII.

Exclusive publishing rights remain with Cahners Publishing Co unless entry is returned to author, or editor gives written permission for publication elsewhere.

In submitting my entry, I agree to abide by the rules of the Design Ideas Program.

Signed _____

Date _____

Your vote determines this issue's winner. Vote now, by circling the appropriate number on the reader inquiry card.