EDITED BY BILL TRAVIS & ANNE WATSON SWAGER

Software PLL locks VCXO to reference

MARTIN OSSMANN, PHILIPS RESEARCH LABS, AACHEN, GERMANY

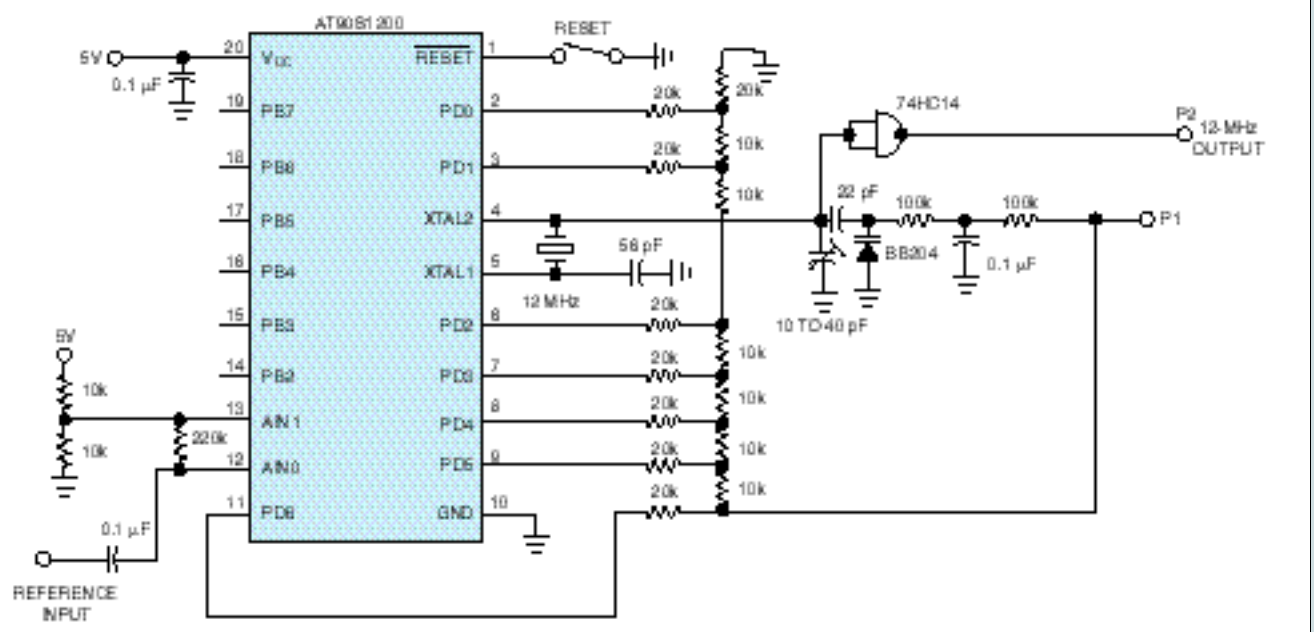
The circuit in **Figure 1** provides a precisely controlled clock signal. The method modifies the controller's quartz oscillator such that the control voltage at point P1 controls the frequency. A D/A converter using an R-2R ladder network connected to output port D generates the control voltage. A precise reference-frequency signal connects to the controller's analog-comparator input, AIN0. In this application, the reference frequency is 162 kHz (derived from a long-wave broadcast station). You can use any other 10- to 200-kHz frequency. The Atmel (www.atmel.com) AT90S1200 controller performs a PLL function to lock the clock to the reference. You can use the precise frequency available at point P2 to clock mCs or other digital circuitry. An 18-instruction program provides the PLL function (**Listing 1**). You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2229.

Figure 2 shows the operating principle. The variables dds0 to dds3 hold a 32-bit, direct-digital-synthesis (DDS)-type numerical oscillator. The routine XORs the

LISTING 1—AT90S1200 CODE FOR PLL FREQUENCY LOCK

```
;   EDN1.ASM:   lock the 12 MHz VCXO to a 162 kHz reference
      .device  at90s1200
      .include "1200def.inc"
.def    tmp     =r16
.def    cntr1   =r17        ; count-and dump time-counter
.def    phase   =r18        ; phase integrator
.def    dds0    =r26        ; LSB of 32 bit DDS register
.def    dds1    =r27
.def    dds2    =r28
.def    dds3    =r29        ; MSB of DDS register

RESET:
        ldi   tmp,$7F       ; port D set to OUTPUT-mode
        out   DDRD,tmp      ; port D is analog 7 bit pase output
SAMPLE:
        mov   tmp,dds3      ; perform DDS-MSB XOR with analog input
        sbic  ACSR,ACO
        com   tmp           ; conditional complement of MSB of DDS
        sbrc  tmp,7         ; conditional increment depending on XOR value
        inc   phase         ; compute phase
noSAMPLE:
        subi  dds0,$fe      ; 32 bit DDS function
        sbci  dds1,$d4      ; DDS rate is 12 MHz / 12 cycles = 1 MHz
        sbci  dds2,$78      ; increment is 162kHz/DDS_rate*2^32
        sbci  dds3,$29      ; is 2978D4FE (hexadecimal)
        subi  cntr1,1       ; integrate-and dump timer
        brne  SAMPLE        ; test for end of count
        clr   cntr1         ; reload timer
        lsr   phase         ; shift phase value to right
        out   PORTD,phase   ; output to 7-Bit DAC
        clr   phase         ; clear phase-value
        rjmp  noSAMPLE      ; and go on, one DDS-cycle is 12 clock periods
```

FIGURE 1

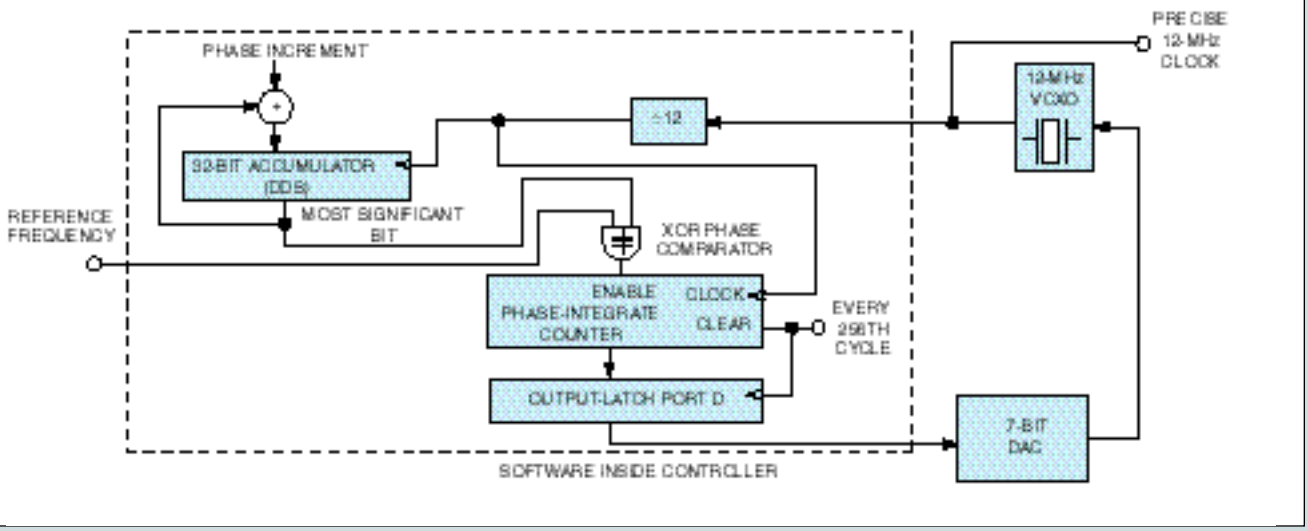


A microcontroller can provide a PLL function to obtain a precise reference-locked frequency.

most significant bit with the reference signal that enters the controller via the analog comparator. A count-and-dump function integrates the XOR output over 255 DDS cycles. Upon every 256th DDS cycle, the phase value routes to the DAC at Port D of the controller. Every loop lasts exactly 12 clock cycles. Thus, the DDS cycle frequency is one-twelfth of the controller's clock frequency. The program performs a phase comparison between the DDS and the reference. The DAC voltage controls the VCXO. By using other increments for the DDS, you can easily adapt the program for other clock or reference frequencies. You can also use the circuit for demodulating phase-modulated signals. (DI #2229) EDN

To Vote For This Design, Circle No. 301

FIGURE 2



The frequency-lock circuit in Figure 1 uses DDS techniques to provide a precise submultiple of the reference frequency.

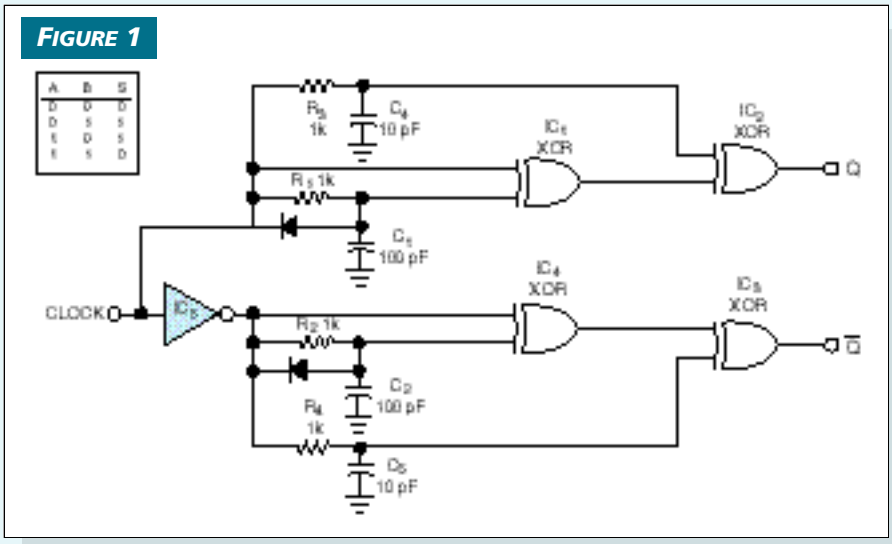
Spice introduces dead time in simulations

CHRISTOPHE BASSO, MOTOROLA SEMICONDUCTOR, TOULOUSE, FRANCE

Bridge or half-bridge designs using MOSFETs or insulated-gate bipolar transistors need some dead time between commutations to avoid any cross-conduction current spikes. This statement is also valid for switch-mode power supplies that use synchronous rectification. In creating simulations, it is sometimes difficult to write the stimuli so as to define a dead time between commutations. Classic PULSE or PWL commands are impractical, especially when either frequency or pulse width changes during the simulation run. **Figure 1** shows a approach to simulating dead time that

An inverter, a few XOR gates, and some passive components generate a dead-time interval for switch commutation.

FIGURE 1

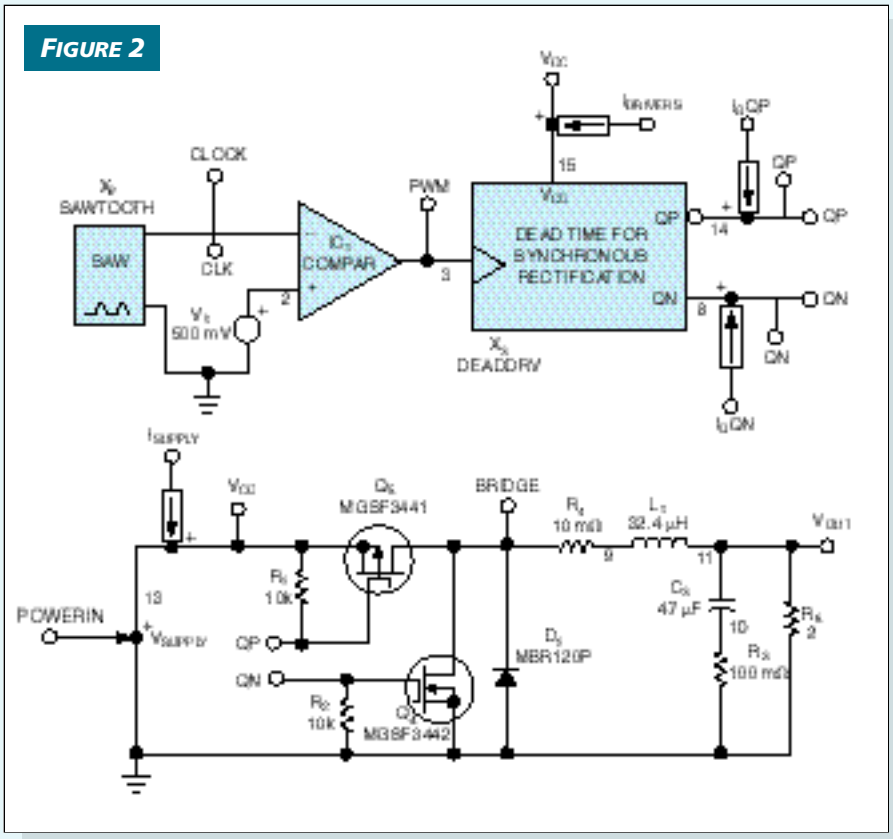


you can build around a few logic XOR gates. The principle uses the truth table of an XOR or XNOR gate: that the output is high or low only when both inputs have different logic states.

The logic states come from the RC networks R_1-C_1 and R_4-C_5. The output of the IC_1 and IC_4 gates is thus a short pulse whose width depends on the RC time constants of the input network. This pulse blanks the signal delivered to the output and thus generates the required dead time. You can easily model the logic functions using Intusoft's (www.intusoft.com) IsSpice4 Analog Behavioral Modeling features (**Listing 1**). You need to feed the subcircuit with the dead-time value as well as the output high and low levels. The input clock is TTL/CMOS-compatible. By changing the B5 line to V=V(26,20)<100MV ? {VHIGH} : {VLOW}, the generator becomes suitable for driving a synchronous rectifier (**Figure 2**). **Figure 3** clearly shows the absence of overlap between commutations. You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DISIG, #2228. (DI #2228) EDN

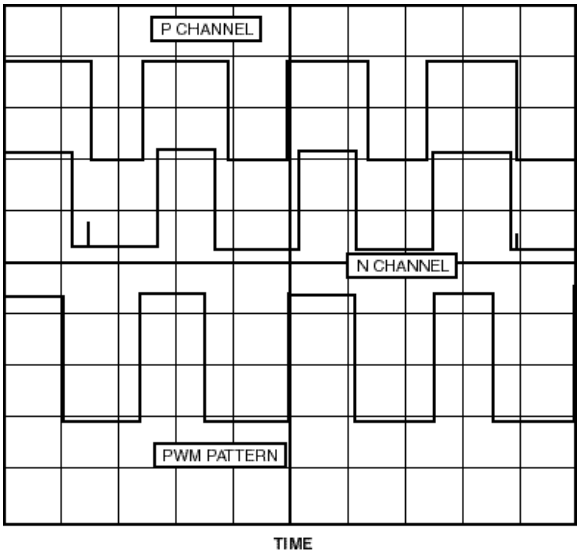
To Vote For This Design, Circle No. 302

FIGURE 2



The circuit in Figure 1 generates the dead time to prevent cross-conduction in this synchronous-rectifier circuit.

FIGURE 3



Current waveforms for the MOSFETs in Figure 2 show no simultaneous conduction.

LISTING 1—ISSPICE4 ANALOG BEHAVIORAL MODELING FEATURES

```
.SUBCKT DEADTIME 1 50 51 {DT=500N VHIGH=10V VLOW=100M RS=10}
* Clock_In Q Qbar
* Developed by Christophe BASSO (FRANCE)
RIN 1 0 1MEG
B6 17 0 V=V(1)>2V ? 10 : 0
R3 17 18 1k
C3 18 0 {DT/(1000*4.14)}
B4 21 0 V=V(25,19)<100MV ? {VLOW} : {VHIGH}
RCQ 21 60 100
CCQ 60 0 10P
BQ 61 0 V=V(60)
RSQ 61 50 {RS}
R4 22 23 1k
C4 23 0 {DT/(1000*4.14)}
B5 24 0 V=V(26,20)<100MV ? {VLOW} : {VHIGH}
RCQB 24 70 100
CCQB 70 0 10P
BQB 71 0 V=V(70)
RSQB 71 51 {RS}
R5 17 25 1k
C5 25 0 {DT/(1000*41.4)}
R6 22 26 1k
C6 26 0 {DT/(1000*41.4)}
D3 23 22 DISCH
D4 18 17 DISCH
B1 22 0 V=V(1)>2V ? 0 : 10
B2 19 0 V=V(17,18)<100MV ? 0 : 10
B3 20 0 V=V(22,23)<100MV ? 0 : 10
.MODEL DISCH D BV=100V CJO=4PF IS=7E-09 M=.45 N=2 RS=.8
+ TT=6E-09 VJ=.6V
.ENDS
```

Circuit translates TTY current loop to RS-232C

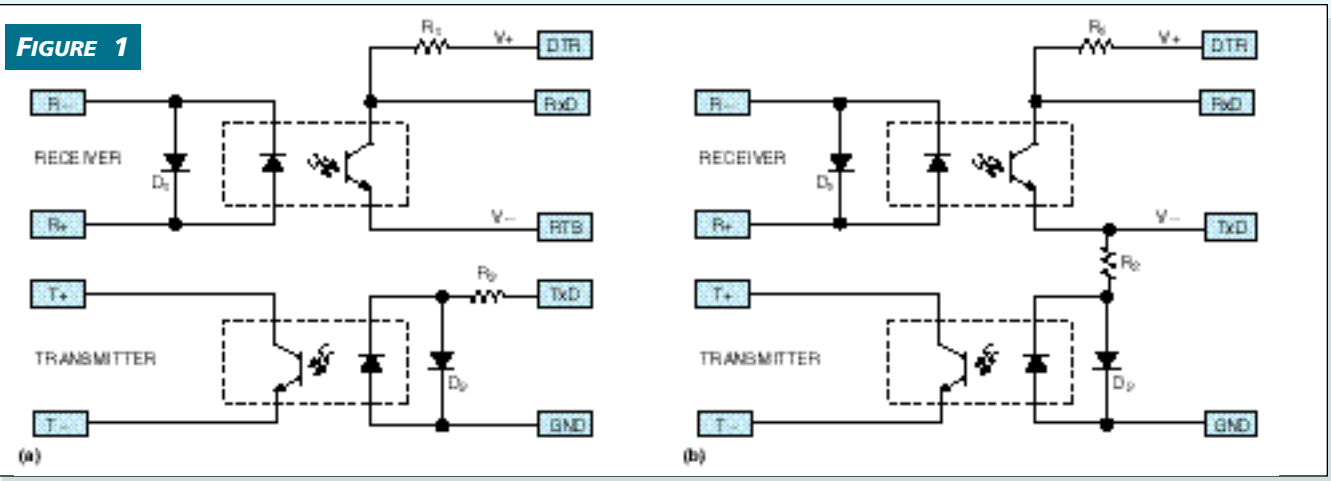
JERZY CHRZASZCZ, WARSAW UNIVERSITY, POLAND

The circuit in **Figure 1a** provides signal translation between a passive current-loop (TTY) interface and a duplex RS-232C port. The current flowing in the receiver loop causes the transistor to pull down RxD; when the transistor turns off, R_1 pulls up RxD. In like manner, the current in the transmitter loop switches on for a negative TxD voltage and off for a positive voltage. The supply power comes from the interface control lines, so you must properly preset these lines. Unfortunately, terminal programs do not usually support direct control of handshake signals. In other words, you must write your own service routine for the serial port to obtain a negative supply.

Worse, some RS-232C-like ports feature just one handshake line, rendering the circuit in **Figure 1a** unusable. In such cases, you could use the interface in **Figure 1b** . Because the transmitter output acts as the negative supply rail, the circuit can receive data only when TxD remains inactive. This limitation obviously precludes full-duplex transmission. Note also that transmitted data directly echoes at the RxD input. However, if half-duplex operation is satisfactory and you can tolerate local echo, this circuit may be the one of choice. In both circuits, external diodes (for example, 1N4148) protect the LEDs against reverse voltages. The values of R_1 and R_2 depend on the optocoupler type and loop current. You can use CNY75B optocouplers with 5.1 and 220V for R_1 and R_2, respectively. (DI #2230) EDN

To Vote For This Design, Circle No. 303

FIGURE 1



An optocoupler and two resistors configure a TTY-to-RS-232C translator. Use the circuit in (a) when a negative supply is available; otherwise, use the circuit in (b).

80186 timer pins provide general I/O

SK SHENOY, NPOL, KOCHI, INDIA

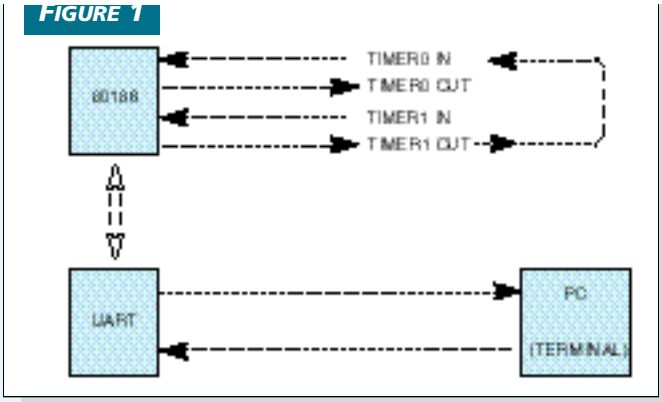
Intel's (www.intel.com) 80186 is a highly integrated mP common in embedded applications. It combines 15 to 20 common iAPX86 system components, such as a DMA controller, an interrupt controller, timers, a clock generator, a bus interface, and chip-select logic on one chip. Unfortunately, unlike many mCs, it provides no general-purpose I/O pins. However, if you require only a couple of input or latched-output lines, you can use the built-in timer I/O pins as general-purpose input or output lines by using this programming method (**Listing 1**).

The 80186 has two timers (0 and 1), each with two external pins: one for input, one for output. However, the structure of the timers allows you to use any pair of pins, either for input or output, one at a time, but not simultaneously. In other words, you cannot use Timer1_Out pin as a latched output and Timer1_In pin as an input at the same time. This limitation exists because, when you use Timer_In for input, the state of Timer_Out may change. Thus, you can simultaneously obtain two latched outputs, two input lines, or one input and one output line. Finally, a lag of a few microseconds occurs for an output to respond, because the output is the result not of a single "Out" instruction but of a sequence of instructions.

Moreover, you can see from the code in **Listing 1** that in some cases one or two timer ticks must elapse before the state changes. The same situation exists in sampling an input level.

```
/*
For compiling and linking; the following make batch file is used.
libs file contains names of library files which are to be linked.
Note that all addresses are system specific.

ic86 %1.c
link86 %1.obj,&<libs
loc86  %1.lnk ss(stack(+4096)) ad(cs(code(01030h),data(05040h))) ST(main)
*/

#pragma ram
#pragma fp
#pragma la
#pragma mod186
#pragma extend

typedef unsigned char byte;
typedef unsigned int word;

/* IC86 Specific header file */
#include  <icc\i86.h>

void delay(word count)
{
     word i;
     for (i=0;i <count;i++);
}

/* System specific addresses and utility routines for serial IO */
#include        <include\addr.h>
#include        <include\SccAsync.c>

/* Starts and stops the counter with Max Reg A in use
   so that timer1_out is High */
void    MakeHi()
{
        outword(0xff5e,0x4004); /* Stop timer 1 */
        outword(0xff58,0x0);    /* Count  = 0 */
        outword(0xff5a,0xffff); /* Large Max Count A */
        outword(0xff5e,0xc000); /* Start Counter, with int clk, A only, No cont 0xc004 */
        outword(0xff5e,0x4004); /* Stop Timer */
}

/* Starts the timer with Max Reg A and B alternating, Max Reg A is
   set for only 1 count and B for a large count. The timer is stopped as
   soon as Max Reg B is in use so that timer1_out is Lo */
void    MakeLo()
{
        outword(0xff5e,0x4003); /* Stop Timer1 */
        if ((inword(0xff5e) & 0x1000) != 0x1000)
        /* If output is now high; ie. A is in use, Make output Lo */
        /* Else output already Low; do nothing */
        {
                outword(0xff58,0x0); /* Count  = 0 */
                outword(0xff5a,0x1); /* Max Count A = 1 */
                outword(0xff5c,0xffff);/* Max Count B = Large value */

                 /* Start Counter, with int clk, Max Reg A&B, not continuous mode */
                 outword(0xff5e,0xc002);
                 /* Wait till B in use ie. Output goes Low */
                 while ((inword(0xff5e) & 0x1000) != 0x1000);
                 outword(0xff5e,0x4002); /* Stop Timer */
        }
}

/* Timer0 is kept running with RTG bit = 0 so that the timer is turned
   on or off depending on whether the Timer0_in is Hi or Lo. This routine
   tests the state of Timer0_in pin by checking if the timer is incrementing */
byte    ReadInput(void)
{
        outword(0xff50,0); /* reset count register */
        delay(3);
        if (inword(0xff50) > 0) /* if counting */
                return(1); /* Timer0_in is High */
        else return(0);
}

main()
{
        byte string[300];
        byte chr;

        init_stack();   /* Assembly routine for initialising stack pointer SP and base SS .*/
        outword(0xff56,0xc001); /* Start timer0 in continuous mode,RTG=0 */
        while (1)
        {
                PrintString("Enter Level to be Output(1/0):");
                chr = GetChar();
                if(chr == '1')
                {
                        PrintChar('1');
                        MakeHi(); /* Output Level 1 */
                }
                else
                {
                        PrintChar('0');
                        MakeLo(); /* Output Level 0 */
                }
                PrintChar(' ');
                PrintString("Input Level is  = ");
                if(ReadInput())
                        PrintChar('1');
                else PrintChar('0');
                crlf();
        }
}
```

FIGURE 1



You can use the timer pins in an 80186 mP to obtain a gener-

TABLE 2—DEFAULT 80186 TIMER REGISTER ADDRESSES

Register name	Timer0	Timer1	Timer2
Mode/control word	FF56H	FF5EH	FF66H
Max count B	FF54H	FF5CH	Not present
Max count A	FF52H	FF5AH	FF62H
Count register	FF50H	FF58H	FF60H

However, in all but the most demanding applications, this delay of a few microseconds should be unobjectionable. For such applications, this technique can save on the additional hardware you need to provide an external port. The demo program uses the setup in **Figure 1** . To output a bit, the routine makes the Timer1_Out pin assume either a 1 or 0 state by activating MaxCount A or B, respectively, and stopping the timer in that state. The state of the Timer_Out pin reflects which Max register the timer uses (register 1 for Max Reg A).

To read an input fed to the Timer0_In pin, the routine keeps Timer0 running. Timer0 is configured so that it counts only if the Timer0_In pin is high. Thus, if Timer0 increments during two consecutive reads separated by a small delay greater than a timer clock period, it means the input level is high. **Listing 1** is a demo C program for obtaining output via the Timer1_Out pin and inputs via the Timer0_In pin. The program sends logic 1 or 0 to the Timer1_Out pin, based on the key (1 or 0) the user presses. If Timer1_Out connects to Timer0_In, as the dashed line in **Figure 1** shows, the program reads and displays the input's logic level through the Timer0_In pin. In the demo system, an RS-232C port on the 80186 hardware, connected to a terminal, serves for user keyboard input.

The demo C program is written and compiled using an Intel IC86 compiler on an 8-MHz 80186 system. However, the same technique is applicable to other mPs/mCs having similar timer capabilities. **Table 1** gives the timer control-word bits. **Table 2** provides the default 80186 timer register addresses.

You can download **Listing 1** from *EDN*'s Web site, www.ednmag.com. At the registered-user area, go into the Software Center to download the file from DI-SIG, #2231. (DI #2231) EDN

To Vote For This Design, Circle No. 304

TABLE 1—TIMER CONTROL-WORD BITS

Bit 0	Set to 1 for continuous-mode running; 0 for one-shot mode
Bit 1	1 for time to alternate between maximum-count register A and B; 0 for A only
Bit 2	1 to select external clock for the timer; 0 for internal, that is, CPU clock/4
Bit 3	If 1, Timer 2 output is used as clock, else internal clock (CPU clock/4) is used
Bit 4	If 0, the input level gates the timer on or off (timer will count for a high)
Bit 5	This is a read-only bit set when the timer reaches its maximum value
Bit 11	This bit has to be 0
Bit 12	This read-only bit indicates which maximum-count register is in use (0 indicates A)
Bit 13	If set, interrupts are generated on every terminal count
Bit 14	If 0, Bit 15 (enable/disable) is ignored, else Bit 15 will take effect
Bit 15	If set, the timer is enabled; 0 stops the timer

Notes: All bits except bits 5 and 12 are read/write.
Bits 5 and 12 are read-only.
Bits not shown don't care.

Linearizing an RTD is easy with Spice

RICHARD FAEHNRICH, CONSULTANT, ARLINGTON HEIGHTS, IL

A popular way to design an accurate temperature probe using an inherently nonlinear device—a resistance temperature detector (RTD)—uses positive multiplicative feedback. However, the transfer function of this method is also nonlinear, making an analytical approach difficult. A simpler method uses Spice to optimize the RTD's response by exciting the sensor with a particular waveform and performing simple signal processing on the circuit's output. You can also measure the output's nonlinearity while quickly trying a range of feedback parameters.

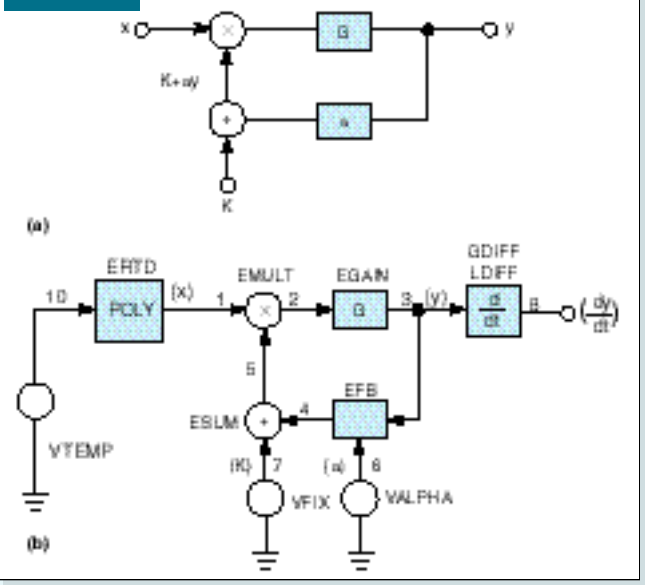
The block diagrams of the multiplicative-feedback circuit (**Figure 1a** and **Reference 1**) show how the scheme multiplies the input by a fraction of the output and a fixed term. The transfer function is

$$y = \frac{x \cdot K \cdot G}{1 - (x \cdot \alpha \cdot G)}.$$

Notice that when the feedback factor, a, is zero, the gain is simply equal to x•K•G. To understand the method, assume initially that x is linear. Introducing a small positive or negative a generates an output that is concave upward or downward, respectively. Now, if the input x exhibits an upward or downward curve, then the proper choice of a bends the curve in the opposite direction to improve the overall linearity.

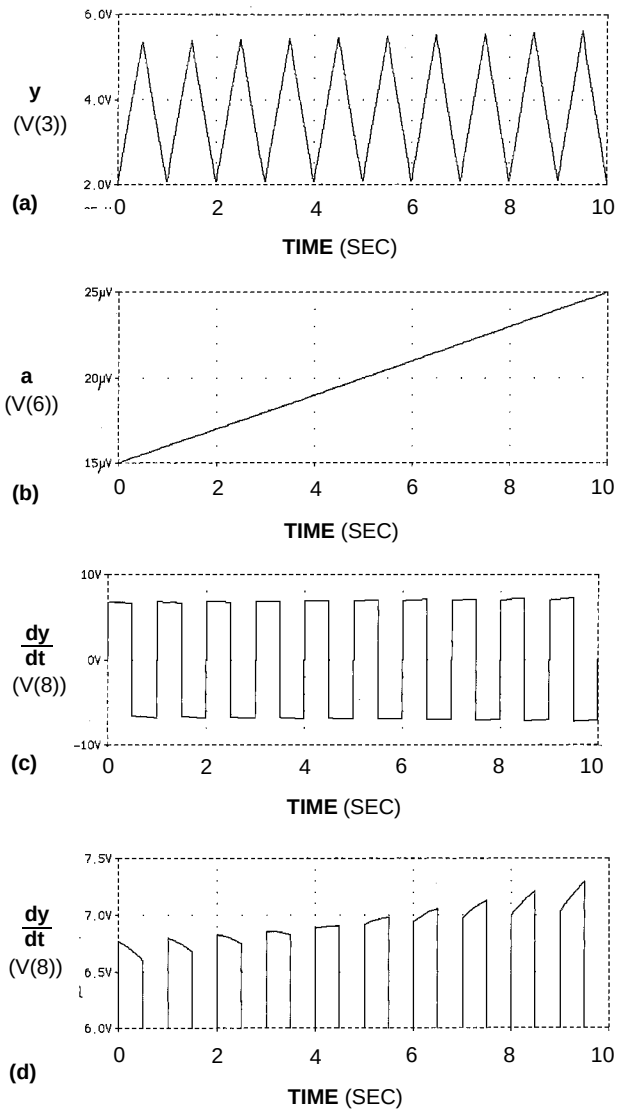
One way to measure the nonlinearity of the output is to fit a straight line to the curve and measure the maximum deviation of the output curve from this line. Another way is to perform a quadratic fit and monitor the second-order coefficient as a measure of straightness. Although both of these methods work, extracting the data from Spice output files

FIGURE 1



Multiplicative feedback (a) is one way to linearize an RTD. You can drive a corresponding Spice circuit (b) with a triangle waveform to measure the circuit's nonlinearity and determine the value of a, for which the derivative of the output is flattest.

FIGURE 2

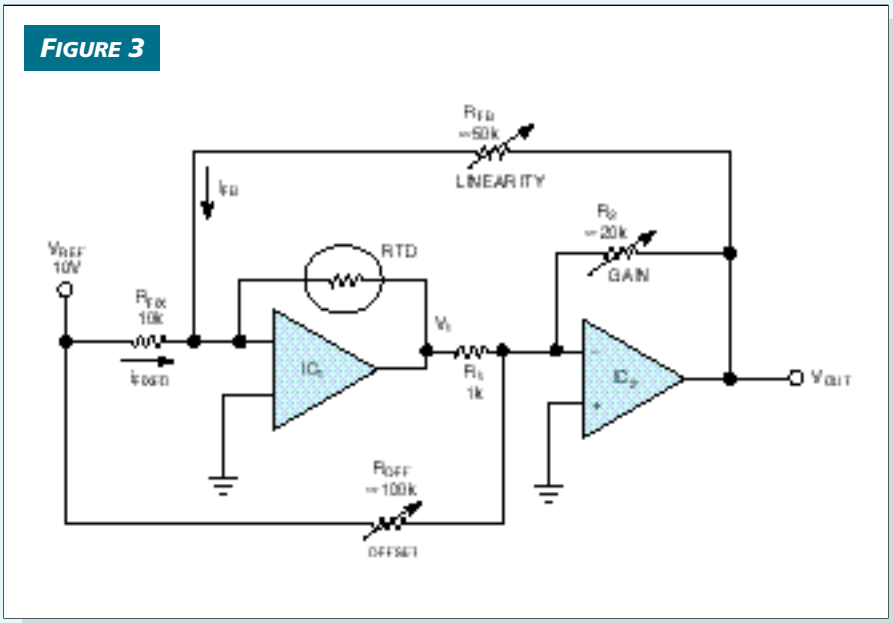


Spice makes it easy to view the output (a), the value of a (b), and the derivative of the output for two horizontal scales (c and d). The optimum value of a is the point at which the derivative is flattest, which occurs approximately 3 to 4 sec into the simulation, or when a '18 mV.

and curve-fitting over a range of values for a are difficult and time-consuming.

Alternatively, you can obtain some measure of nonlinearity by sweeping the temperature with a linear ramp function and then taking a derivative of the output response. A linear output should produce a constant derivative, and a concave upward or downward response produces an increasing or decreasing derivative, respectively. Therefore, the goal of this optimizing method is to find the value of a that results in the flattest derivative. However, because the input-ramp function increases without bound, this method instead uses a triangle wave to sweep the temperature. Now, by slowly sweeping a over many cycles of the triangle wave, a single simulation can measure the circuit's performance with varying parameters.

FIGURE 3



A practical implementation of the RTD linearizing circuit includes linearity, gain, and offset adjustments.

In the Spice circuit, VTEMP drives the input temperature linearly from 0 to 4008C with a 1-Hz triangle waveform (**Figure 1b**). (You can download the corresponding Spice netlist from *EDN*'s Web site, www.ednmag. com. At the registered-user area, go into the "Software Center" to download the file from DI-SIG, #2224.) ERTD simulates the RTD's response versus temperature using a second-order polynomial. To counteract the RTD's concave-downward behavior, the circuit introduces a positive feedback factor, a, with a concave-upward characteristic. VALPHA sweeps a linearly for 10 sec. Therefore, the simulation determines the performance of the circuit for the a range over 10 cycles of the triangle wave. The sum of the positive-feedback term, EFB, and a fixed term, VFIX, controls the gain of the multiplying block, EMULT. EGAIN provides a gain of G. Current source GDIFF and inductor LDIFF form the differentiator.

The linearized output and the a parameter appear in **Figure 2a** and **b**, respectively. **Figure 2c** shows the output derivative, and **Figure 2d** shows only the tops of the derivative's positive cycles. Note how the derivative in **Figure 2d** flattens, surrounded by a decreasing and increasing derivative to the left and right of the optimal behavior, respectively. You can start with a broad range for a and then quickly narrow the search after several simulations. Note that sweeping alpha during the triangle-wave cycles has some influence on the output response and, consequently, its derivative. However, this effect lessens as you narrow the range of a.

You can implement the positive-feedback factor as a potentiometer with an adjustment range that brackets the optimum value as determined by your simulations. Note that the overall gain and offset of the output versus temperature vary as a varies. Therefore, the final circuit should include a gain and offset adjustment in addition to the linearity adjustment.

In one suggested circuit implementation, you repeat the three adjustments until the circuit achieves the correct output voltages for the RTD resistances corresponding to the low, midpoint, and high calibration temperatures (**Figure 3** and **Reference 2**). This circuit realizes the fixed gain and feedback terms as currents. Op amp IC_1 implements the multiplier by first summing the fixed and feedback currents and then multiplying the result by the RTD's resistance. IC_2 provides additional gain, G, producing an output voltage given by

$$V_{OUT} = R_{RTD} \langle (i_{FIXED} + i_{FB}) \langle G,$$

where $G=R_2/R_1$. The feedback-factor calculation is

$$\alpha = \frac{i_{FB}}{V_{OUT}} = \frac{1}{R_{FB}}.$$

A feedback factor of a =0.00002 (R_{FB}=50 kV) in **Figure 3** reduces the linearity error from ±3.08C to less than ±0.18C over a 4008C range. (DI #2224) EDN

References

1. Sheingold, D, "Transducer Interfacing Handbook," Analog Devices, Norwood, MA, 1980, pg 99.

2. "The Linear Applications Handbook," 1990, Linear Technology, Application Note 6, pg 2.

To Vote For This Design, Circle No. 305

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 02458

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.

CIRCLE NO. 10