An Introduction to Acoustic Thermometry

An Air Filled Olive Jar Teaches Signal Conditioning

Jim Williams and Omar Sanchez-Felipe

Introduction

We occasionally lecture to university engineering students. A goal of these lectures is to present technology in a novel, even charming, way. This hopefully entices the student towards the topic; an aroused curiosity is fertile ground for education. One such lecture investigates acoustic thermometry as an example of signal conditioning techniques. This subject has drawn enough interest that it is presented here for wider dissemination and as supplementary material for future acoustic thermometry lectures.

Acoustic Thermometry

Acoustic thermometry is an arcane, elegant temperature measurement technique. It utilizes sound's temperature dependent transit time in a medium to measure temperature. The medium may be a solid, liquid or gas. Acoustic thermometers function in environments that conventional sensors cannot tolerate. Examples include extreme temperatures, applications where the sensor would be subjected to destructive physical abuse and nuclear reactors. Gas path acoustic thermometers respond very quickly to temperature changes because they have essentially no thermal mass or lag. An acoustic thermometer's "body" *is* the measurand. Additionally, an acoustic thermometer's reported "temperature" represents the total measurement path transit time as opposed to a conventional sensor's single point determination. As such, an acoustic thermometer is blind to temperature variations within the measurement path. It reports the measurement path's delay as its "temperature," whether or not the path is iso-thermal.

A pleasant surprise is that the sonic transit time in a gas path thermometer is almost entirely insensitive to pressure and humidity, leaving temperature as the sole determinant. Additionally, sonic speed in air varies predictably as the square root of temperature.

Practical Considerations

A practical acoustic thermometer demonstration begins with selecting a sonic transducer and a dimensionally stable measurement path. A wideband ultrasonic transducer is desirable to promote fast, low jitter, hi-fidelity response free of resonances and other parasitics. The electrostatic type specified in Figure 1 meets these requirements. A single transducer serves as both transmitter and receiver.

, LT, LTC, LTM, Linear Technology and the Linear logo are registered trademarks of Linear Technology Corporation. All other trademarks are the property of their respective owners.

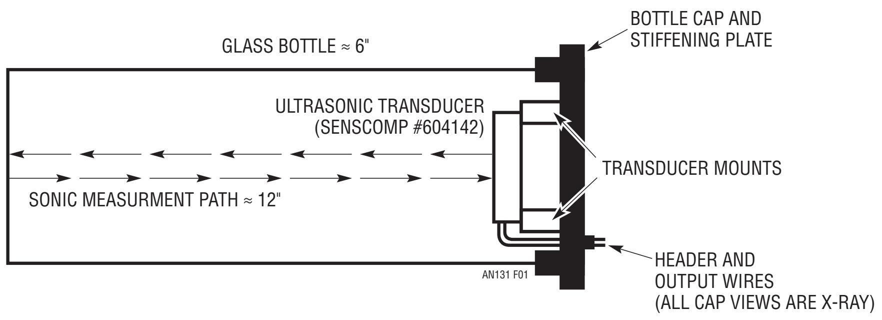
GLASS BOTTLE ≈ 6"
BOTTLE CAP AND STIFFENING PLATE
ULTRASONIC TRANSDUCER (SENSCOMP #604142)
TRANSDUCER MOUNTS
SONIC MEASURMENT PATH ≈ 12"
HEADER AND OUTPUT WIRES (ALL CAP VIEWS ARE X-RAY)
AN131 F01

Figure 1. Ultrasonic Transducer Rigidly Mounts Within Stiffened Cap Affixed to Bottle. Structure Defines Fixed Length Measurement Path Essentially Independent of Physical Variables. Sonic Transit Time at 75°F ≈ 900µs with ≈1µs/°F Variation

The device is rigidly mounted within the stiffened metal cap of a glass enclosure, promoting measurement path dimensional stability. The enclosure and its cap are conveniently furnished by a bottle of "Reese" brand "Cannonball" olives (Figure 2). After removing the olives and their residue, the bottle and cap are baked out at 100°C prior to joining. The transducer leads pass through the cap via a coaxial header. This arrangement, along with the glass enclosure's relatively small thermal expansion coefficient, yields a path distance stable against temperature, pressure and mechanically induced changes. Path length, including enclosure bottom bounce and return to transducer, is about 12". This sets a two-way trip time of around 900µs (speed of sound in air ≈ 1.1 ft/ms). At 75°F, this path's temperature dependent variation is approximately 1µs/°F. A desired 0.1°F resolution mandates mechanical and electronic induced path length uncertainty inside 100ns; approximately 0.001" dimensional stability referred to the 12" path length. Considering likely error sources, this is a realistic goal.

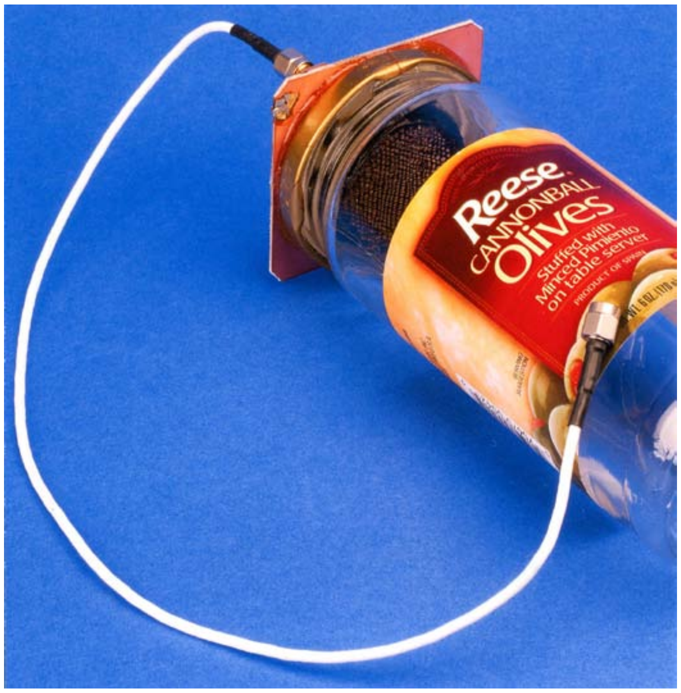

Figure 2. Photograph Details Cap Assembly, Partially Views Glass Enclosed Measurement Path. Ultrasonic Transducer Visible Within Cap. Stiffening Plate, Bonded to Cap Top, Prevents Ambient Pressure or Temperature Changes from Deforming Thin Metal Cap, Promoting Measurement Path Length Stability. Coaxial Header Provides Transducer Connections

Overview

Figure 3 is a simplified overview of the acoustic thermometer. The transducer, which can be considered a capacitor, is biased at $150V_{DC}$. The start pulse clock drives it with a short impulse, launching an ultrasonic event into the measurement path. Simultaneously, the width decoding flip-flop is set high. The sonic impulse bounces off the enclosure bottom, returns to the transducer and impinges on it. The resulting minuscule mechanical displacement causes the transducer to give up charge ($Q = \Delta C \bullet V$), which appears as a voltage at the receiver amplifier input. The trigger converts the amplifier's output excursion into a logic compatible level which resets the flip-flop. The flip-flop output width represents the measurement path's temperature dependent sonic transit time. The microprocessor, equipped with the measurement path's temperature/delay calibration constants, calculates the temperature and supplies this information to the display.[1] The start pulse generator has a second output which gates the trigger output off during nearly the entire measurement cycle. The trigger output only passes during the immediate time vicinity when a return pulse is expected. This discriminates against unwanted sonic events originating outside the measurement path, eliminating false triggers. A second gating, sourced from the width decoding flip-flop, shuts down the 150V bias supply switching regulator during the measuring interval. Return pulse amplitude is under 2mV at the transducer and the high gain, wideband receiver amplifier is vulnerable to parasitic inputs. Shutting down the 150V bias supply during the measurement prevents its switching harmonics from corrupting the amplifier. Figure 4 describes the system's event sequence. A measurement cycle begins with a start pulse (A) driving the transducer and setting the flip-flop (B) high. After the sonic impulse's transit time, the amplifier responds (C, diagram right), tripping the trigger (D, diagram right) which resets the flip-flop. Gate signals (E and F) protect the trigger from unwanted sonic events and start pulse artifacts and shut off the high voltage regulator during measurement.

Note 1. See Appendix A, "Measurement Path Calibration" for details on determining calibration constants.

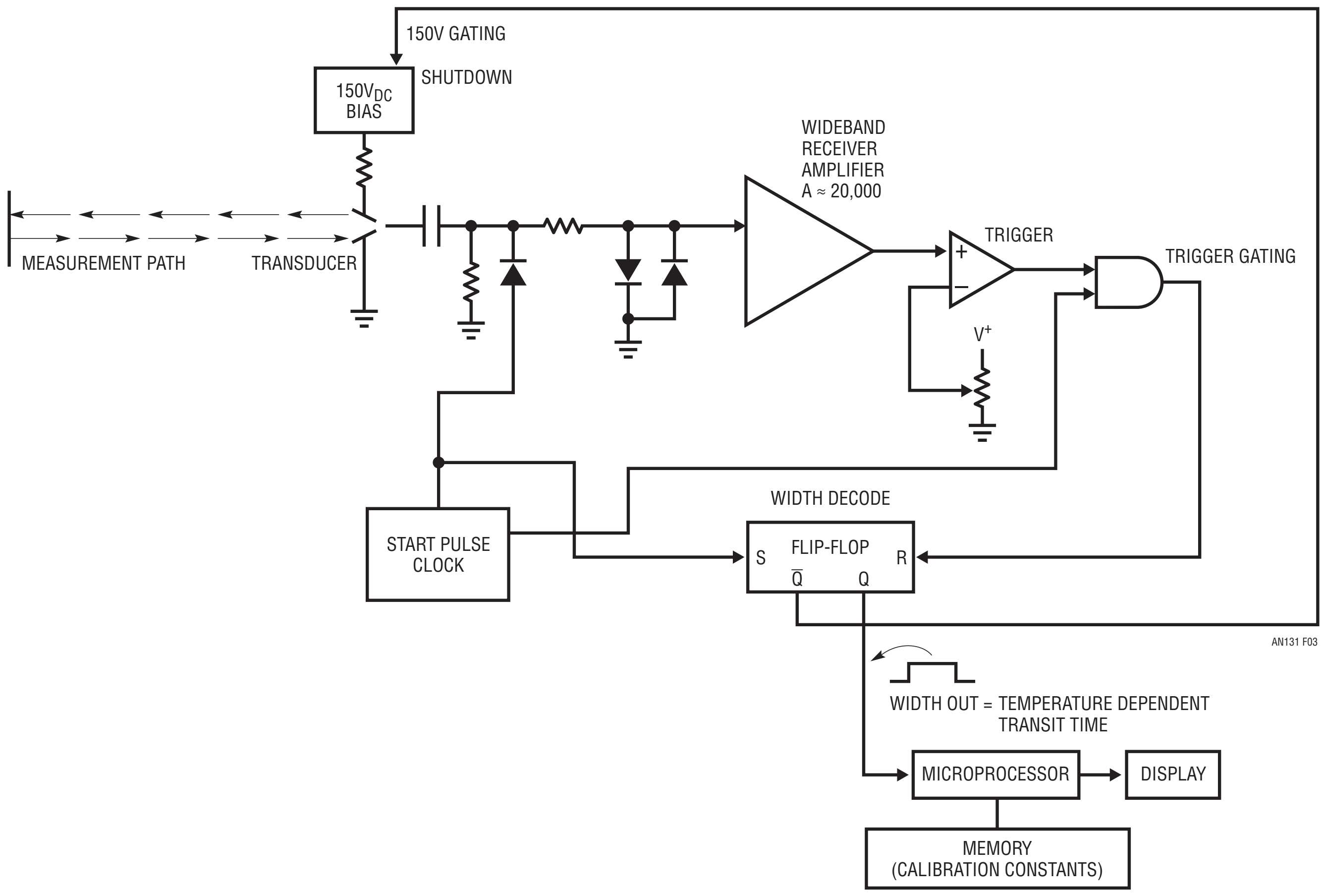

Figure 3. Conceptual Signal Conditioning for Acoustic Thermometer. Start Clock Launches Acoustic Pulse into Measurement Path, Sets Width Decode Flip-Flop High. Acoustic Paths Return Pulse, Amplified by Receiver, Trips Trigger, Resetting Flip-Flop. Resultant "Q" Width Output, Representing Path's Temperature Dependent Transit Time, is Converted to Temperature Reading by Microprocessor. Gating High Voltage Supply and Trigger Prevents Spurious Outputs

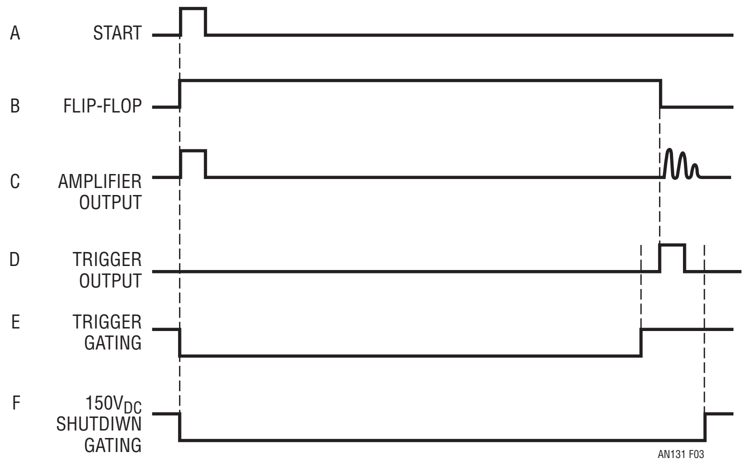

Figure 4. Figure 3's Event Sequence. Start Pulse (A) Drives Transducer, Sets Flip-Flop (B) High. Sonic Pulse Return Activates Amplifier (C, Extreme Right), Causing Trigger Output (D) to Reset Flip-Flop (B). Trigger Gating (E) Prevents Erroneous Trigger Response to Start Pulse Caused Amplifier Output (C, Extreme Left) and External Sonic Events. Gating (F) Turns Off 150V Switching Converter During Measurement, Precluding Amplifier Output Corruption

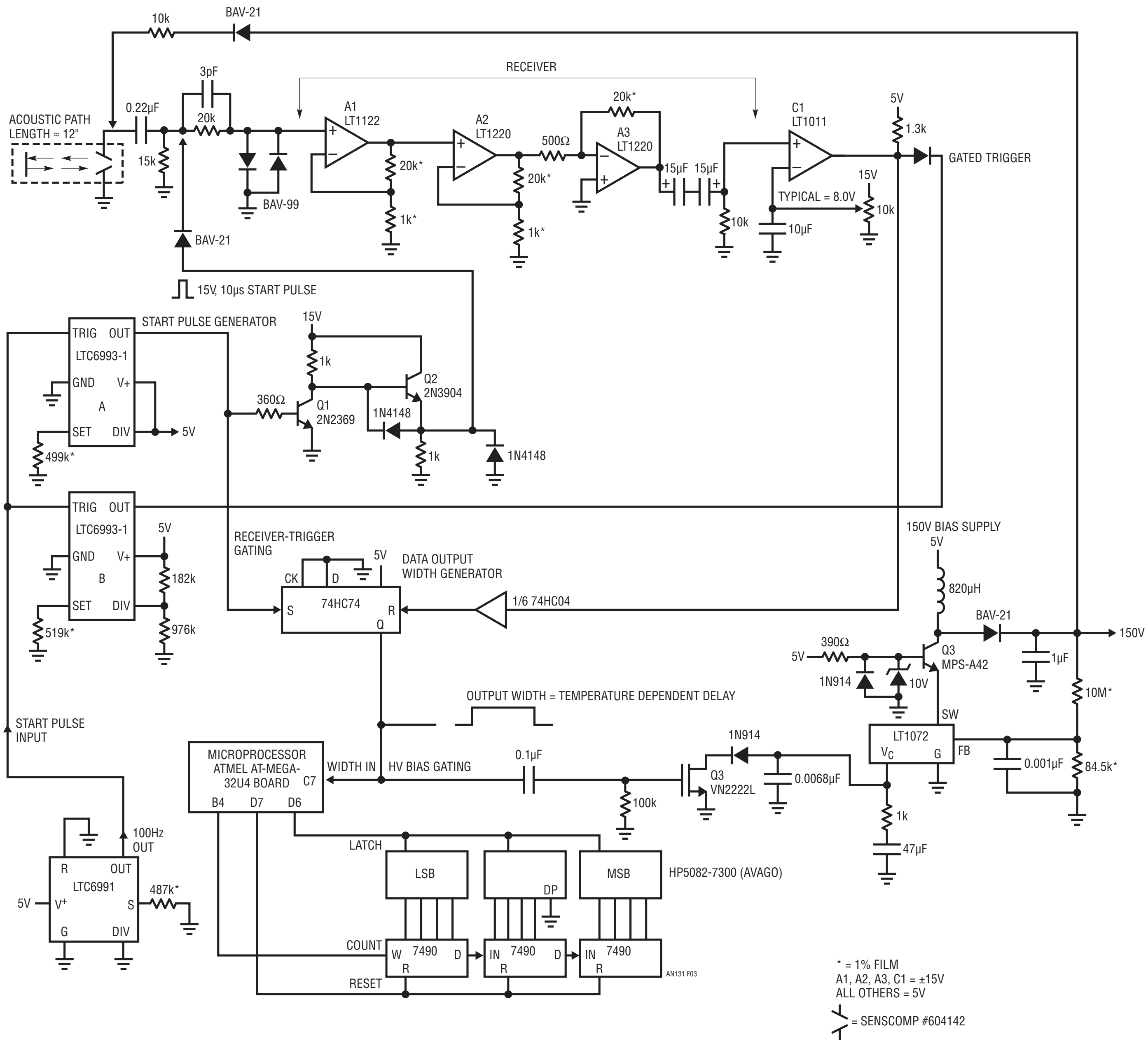

Figure 5. Detailed Circuitry Closely Follows Figure 3's Concept. Start Pulse Generator is Comprised of 100Hz Clock, One-Shot Multivibrators and Q1-Q2 Driver Stage. A ≈ 20,000 Receiver Amplifier Splits Gain Among Three Stages, Biases Trigger Comparator. Flip-Flop Output Width Feeds Microprocessor Which Calculates and Displays Temperature. Capacitive Coupling Isolates High Voltage DC Transducer Bias, Diode Clamping Prevents Destructive Overloads. Switching Regulator Controls High Voltage Via Cascode. Gating Obviates Switching Regulator Noise Originated Interference, Minimizes External Sonic Corruption

Detailed Circuitry

Figure 5's detailed schematic closely follows Figure 3's concepts. An LTC®6991 oscillator furnishes the 100Hz clock. LTC®6993-1 monostable "A" provides a 10μs width to the Q1-Q2 driver, which capacitively couples the start pulse (Trace A, Figure 6) to the transducer. Simultaneously, the monostable sets the flip-flop (E) high. The flip-flop high output shuts down the LT®1072 based high voltage converter during the measurement. Monostable "B" produces a pulse (B) which gates off C1's trigger output for a time just shorter than the fastest expected sonic return.

The launched sonic pulse travels down the measurement path, bounces, returns, and impinges on the transducer. The transducer, biased at $150V_{DC}$, releases charge ($Q = \Delta C \bullet V$) which appears as a voltage at the receiver amplifier. The cascaded amplifier, with an overall gain of ≈ 17,600, produces A2's output (C) and a further amplified version at A3. C1 triggers (D) at the first event that exceeds its negative input threshold, resetting the flip-flop. The resultant flip-flop width, representing the temperature dependent transit time, is read by the microprocessor which determines the temperature and displays it.[2]

Figure 7 studies receiver amplifier operation at the critical return impulse trip point. The returning sonic pulse is viewed at A2 (Trace A). A3 (B) adds gain, softly saturating the signals leading response. C1's trigger output (C) responds to multiple triggers but the flip-flop output (D) remains high after the initial trigger, securing transit time data.

Note 2. Complete processor software code appears in Appendix B, "Software Code."

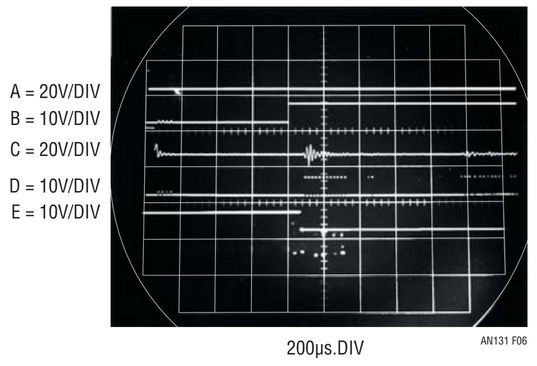

Figure 6. Figure 5's Waveforms Include Start Pulse (A), Trigger Gate (B), A2 Output (C), Trigger (D) and Flip-Flop Q Output Width (E). Amplifier Output Causes Multiple Trigger Transitions, But Flip-Flop Maintains Pulse Width Integrity. A2-Trigger Outputs at Photo Right, Due to Acoustic Pulse Second Bounce, Are Inconsequential

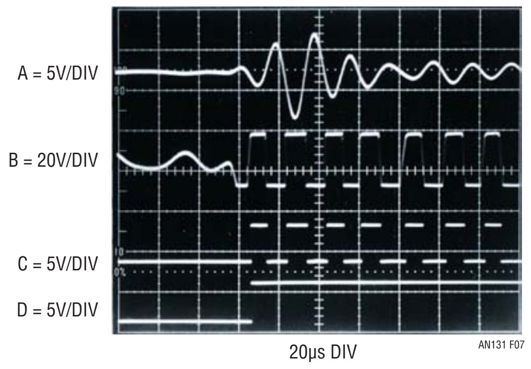

Figure 7. Receiver Amplifier—Trigger Operating Detail at Trip Point. Returning Sonic Pulse Viewed at A2 Output (Trace A) After X440 Gain. A3's Output (B) Adds X40, Softly Saturating Signals Leading Response. C1's Output (C) Responds to Multiple Triggers But Flip-Flop Output (D) Remains High After Initial Trigger

Gating prevents high voltage supply switching harmonics from producing spurious amplifier-trigger outputs. Figure 8 shows gate-off detail. The flip-flop output (Trace A) going high shuts down high voltage switching (B) at the measurement onset. This state persists during the entire transit time, preventing erroneous amplifier-trigger outputs. Figure 9's flip-flop fall (Trace A) combines with LT1072 V_C pin associated components, producing delayed high voltage turn-on (B) after the vulnerable, small amplitude return pulse trip point. This assures a clean, noise-free trigger.

Several circuit attributes aid performance. As mentioned, gating the trigger output prevents sonic interference from outside sources. Similarly, gating the 150V converter off prevents its harmonics from corrupting the high gain, wideband receiver amplifier. Additionally, the 150V supply value is a gain term, making its regulation loss during the measurement a potential concern. Practically, the 1µF output capacitor decays only 30mV in this time, or about 0.02%. This small variation is constant, insignificant and may be ignored. Deriving the trigger trip point and the start pulse from the same supply allows trigger voltage to vary ratiometrically with received signal amplitude, enhancing stability. Finally, the transducer used is wideband, highly sensitive and free from resonances, promoting repeatable, jitter-free operation.[3] All of the above directly contribute to the circuit's < 100ns (0.1°F) resolution of the ≈ 1ms path length – less than 100ppm uncertainty. Absolute accuracy from 60°F to 90°F is within 1°F referenced to Appendix A's calibration.

Note 3. Readers rich in years will recognize the specified transducer descends from 1970's era Polaroid SX-70 automatic focus cameras (every baby boomer had to have one).

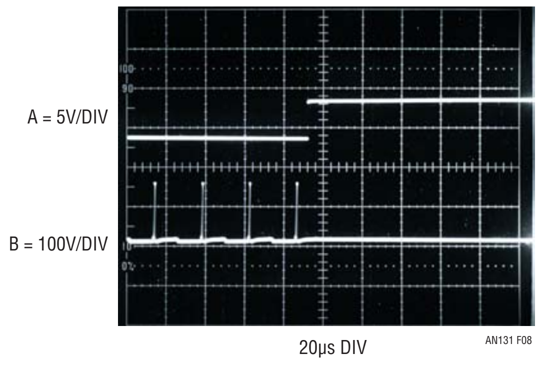

Figure 8. High Voltage Bias Supply Gate-Off Detail. Flip-Flop Output (Trace A) Going High Shuts Down LT1072 Switching Regulator, Turning Off 150V Flyback Events (B). Off-Time Extends During Measurement Interval, Precluding Receiver Amplifier Corruption

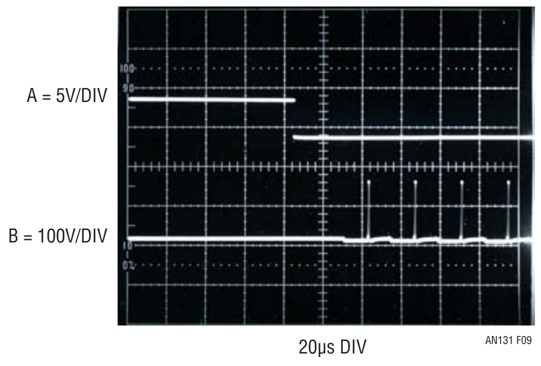

Figure 9. High Voltage Bias Supply Gate-On Detail. Flip-Flop Output (Trace A) Drops Low at Sonic Pulse Return. Components at LT1072 Switching Regulator V_C Pin Delay Bias Supply Turn-On (B). Sequencing Ensures Bias Supply is Off During Amplifier-Trigger's Vulnerable Response to Sonic Pulse Arrival

Triggering on later bounces offers the potential benefit of easing timing tolerances and merits consideration. Figure 10 shows multiple sonic bounces, discernible in A2's output, decaying into acoustic dispersion induced noise contained within the glass enclosure. Triggering on a later bounce would relax timing margins, but incurs unfavorable signal-to-noise characteristics. Signal processing techniques could overcome this, but the increased resolutions utility would have to justify the effort.

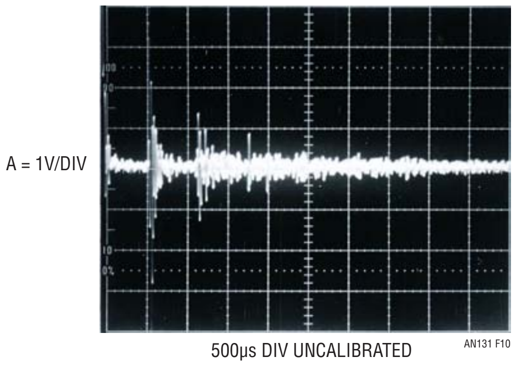

Figure 10. Multiple Sonic Bounces, Discernable in A2's Output, Decay into Acoustic Dispersion Induced Noise Contained Within Glass Enclosure. Triggering on Later Bounce Would Ease Timing Tolerances, But Incur Signal-to-Noise Degradation

REFERENCES

1. Lynnworth, L.C. and Carnevale, E.H., "Ultrasonic Thermometry Using Pulse Techniques." *Temperature: Its Measurement and Control in Science and Industry*, Volume 4, p. 715-732, Instrument Society of America (1972).

2. Mi, X.B. Zhang, S.Y. Zhang, J.J. and Yang, Y.T., "Automatic Ultrasonic Thermometry." Presented at Fifteenth Symposium on Thermophysical Properties, June 22-27, 2003, Boulder, Colorado, U.S.A.

3. Williams, Jim, "Some Techniques For Direct Digitization of Transducer Outputs," Linear Technology Corporation, Application Note 7, p.4-6, Feb. 1985.

4. Analog Devices Inc., "Multiplier Applications Guide," "Acoustic Thermometer," Analog Devices Inc., p. 11-13, 1978.

APPENDIX A

Measurement Path Calibration

Theoretically, temperature calibration constants can be calculated from measurement path length. In practice, it is difficult to determine path length to the required accuracy. Enclosure, transducer and mounting dimensional uncertainties necessitate calibration vs known temperatures. Figure A1 shows the calibration arrangement. The enclosure is placed in a controllable thermal chamber equipped with an accurate thermometer iso-thermal with the enclosure. Ten evenly spaced temperature points from 60°F to 90°F are generated by stepping chamber set-point. The enclosure has a 30 minute time constant to 0.25°F settling, so adequate time must be allowed for each step to stabilize before readings are taken. Readings consist of noting the counter indicated pulse width at each temperature and recording the data. This information is then loaded into the microprocessor memory. See Appendix B, "Software Code."

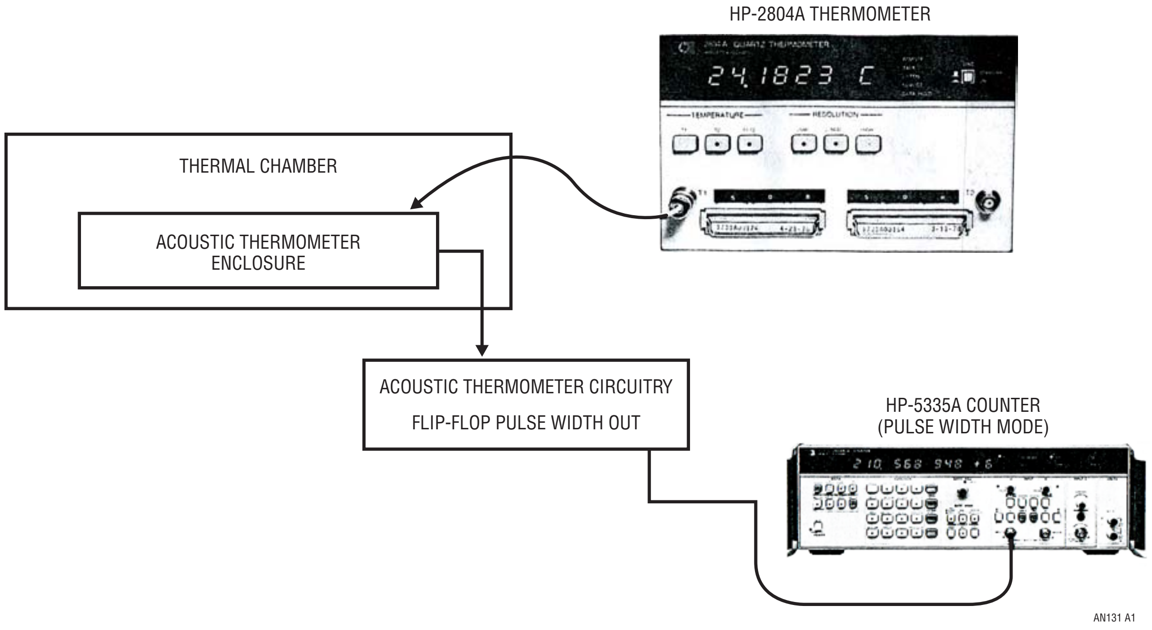
HP-2804A THERMOMETER
24.1823 C
THERMAL CHAMBER
ACOUSTIC THERMOMETER ENCLOSURE
ACOUSTIC THERMOMETER CIRCUITRY
FLIP-FLOP PULSE WIDTH OUT
HP-5335A COUNTER (PULSE WIDTH MODE)
210.568 948 +6
AN131 A1

Figure A1. Calibration Arrangement Consists of Thermometer, Counter, Acoustic Conditioning Circuitry and Enclosure in Thermal Chamber. Temperature is Stepped Every 3°F Between 60°F and 90°F Allowing 30 Minute Stabilization Time Per Point

APPENDIX B

The software code for the Atmel AT-Mega 32U4 microprocessor, combined with the calibration constants stored in its memory (see Appendix A), enables the processor to calculate and display the sensed temperature. This code, written by Omar Sanchez-Felipe of LTC, appears below.

```
//      OLIVER.C:
//
//      Hot temps from a jar of olives.
//
//      The microprocessor is the Atmel ATmega32U4. See 'atmel.com' for its
//      datasheet. The code is compiled with the current version of the
//      'avr-gcc' compiler obtainable at 'winavr.sourceforge.net'.
//
//      Calibration data:
//          Pulse len (usecs)        Temp (deg)
//           877.40                   84.2
//           884.34                   76.0
//           887.40                   72.4
//           892.56                   66.4
//           897.60                   60.7
//
#include <avr/io.h>
#include <avr/wdt.h>

//    Some useful defs
#define  BYTE             unsigned char
#define  WORD             unsigned short
#define  DWORD            unsigned long
#define  BOOL             unsigned char
#define  NULL             (void *)0
#define  TRUE             1
#define  FALSE            0

#define  BSET(r, n)       r |= (1<<n)        // set/clr/tst nth bit of reg
#define  BCLR(r, n)       r &= ~(1<<n)
#define  BTST(r, n)       (r & (1<<n))
#define  DDOUT(r, v)      (r |= (v))         // config (set) bits for OUT
#define  DDIN(r, v)       (r &= ~(v))        // config (clear) bits for IN
#define  SETBITS(r, v)    (r |= (v))         // set multiple bits on register
#define  CLRBITS(r, v)    (r &= ~(v))        // clear ""

#define  SYS_CLK          16000000L          // clock (Hz)
#define  CLKPERIOD        625
#define  POSEDGE          1                  // edge definitions for timerwait
#define  NEGEDGE          0
#define  TMRGATE          PORTC7             // I/O pins
#define  BCDRSET          PORTD7
#define  BCDPULS          PORTB4
#define  DPYLATCH         PORTD6
```

```
//      Calibration table and entry.
//      Using small units of time and temp allows all calcs
//      to be done in fixed point.
struct  calpoint
{
        DWORD pulse;          // pulse duration, in tenths of nsecs
        WORD  temp;           // temperature, in tenths of degrees
        WORD  slope;          // slope (in tenths of nsecs/tenths of degree)
};

struct  calpoint caltab[] =   // the calibration table
{       {8774000L, 842, 0},
        {8843400L, 760, 0},
        {8874000L, 724, 0},
        {8925600L, 664, 0},
        {8976000L, 607, 0}
};
#define NCALS        (sizeof(caltab)/sizeof(struct calpoint))
#define TEMPERR      999

void    spin(WORD);
BOOL    timerwait(BYTE v, long tmo, WORD *p);
void    setdpy(WORD);
void    dobackgnd(void);
WORD    dotemp();

int     main()
{
        WORD  temp, i;

        // set master clock divisor
        CLKPR = (1<<CLKPCE); CLKPR = 0;

        // clear WDT
        MCUSR = 0;
        WDTCSR |= (1<<WDCE) | (1<<WDE);
        WDTCSR  = 0x00;

        // init the display I/O pins
        BCLR(PORTD, BCDRSET);
        BCLR(PORTD, DPYLATCH);
        DDOUT(DDRD, ((1<<BCDRSET) | (1<<DPYLATCH)));
        BCLR(PORTB, BCDPULS);
        DDOUT(DDRB, (1<<BCDPULS));

        // init pulse-width counter (timer #3)
        TCCR3A = 0x00;
        TCCR3B = 0x01;              // no clk prescaling
        DDIN(DDRC, (1<<DDC7));      // PC7 is gate
        BSET(PORTC, PORTC7);        // enable pullup

        // compute the slope for each entry in the cal table
        // the first slope is never used, so we leave it at 0
        // note we're inverting the sign of the slope
        for (i = 1; i < NCALS; i++)
              caltab[i].slope =    (caltab[i].pulse - caltab[i-1].pulse) /
                                   (caltab[i-1].temp - caltab[i].temp);
        setdpy(0);
for (;;)
        {
        temp = dotemp();     // compute temperature
        setdpy(temp);        // set the display
        spin(1000);          // spin a second and repeat
        }
} // main

//      Set the display LED’s to specified count by brute-force
//      incrementing the BCD counter that feeds it.
//
void    setdpy(WORD cnt)
{
        // reset BCD counter
        BSET(PORTD, BCDRSET);
        asm(“nop”); asm(“nop”);
        BCLR(PORTD, BCDRSET);

        while (cnt--)
              {
              BSET(PORTB, BCDPULS);
              asm(“nop”); asm(“nop”);
              BCLR(PORTB, BCDPULS);
              asm(“nop”); asm(“nop”);
              }
        // latch current val (to avoid flicker)
        BCLR(PORTD, DPYLATCH);
        asm(“nop”); asm(“nop”);
        BSET(PORTD, DPYLATCH);
}

//      Set up the edge detector to trigger on either a positive or
//      negative edge, depending on the ‘edge’ flag, and then wait for
//      it to happen. See defines for POSEDGE/NEGEDGE above.
//      if param ‘tmo’ is TRUE, a timeout is set so that we dont wait
//      forever if no pulse materializes.  Returns the counter value
//      at which the edge occurs via pointer ‘p’.
//
#define tcnTMO(SYS_CLK/20000L)       // approx 1 msecs
BOOL    timerwait(BYTE edge, long tmo, WORD *p)
{
        if (edge == NEGEDGE)
              BCLR(TCCR3B, ICES3);   // falling edge
        else  BSET(TCCR3B, ICES3);   // rising edge
        BSET(TIFR3, ICF3);
        tmo *= tcnTMO;
        while (!BTST(TIFR3, ICF3))
              {if (tmo-- < 0)
                    return FALSE;
              }
        *p = ICR3;   // return current counter
        return TRUE;
}
```

```
//      Measure the next POSITIVE pulse and map into temperature.
//
WORD    dotemp()
{
        WORD  strt, end, i;
        DWORD dur, temp;

        strt = end = dur = 0;
        // wait for any ongoing pulse to complete
        if (!timerwait(NEGEDGE, 5000, &strt))
              return 0;
        // now catch the first positive pulse
        if (!timerwait(POSEDGE, 5000, &strt))
              return 0;
        // and wait for it to complete
        if (!timerwait(NEGEDGE, 5000, &end))
              return 0;
        dur = (end - strt);
        dur *= CLKPERIOD;  // duration now in tenths of nsecs

        // compute temp. If warmer than highest calibrated temp
        // or cooler than lowest calibrated temp, return TEMPERR
        //
        if (dur < caltab[0].pulse || dur > caltab[NCALS-1].pulse)
              temp = TEMPERR;
        else  { // an entry will always be found, but we (re)init
                // temp to avoid complaints from the compiler
              temp = TEMPERR;
              for (i = 1; i < NCALS; i++)
                    {if (dur <= caltab[i].pulse)
                          {temp = caltab[i].temp +
                                (caltab[i].pulse - dur) /
                                      caltab[i].slope;
                          break;
                          }
                    }
              }
        return temp;
}

//      ----------------------------------------------------------------
//      Spin for 'ms' millisecs. Spin constant is empirically determined.
//
#define SPINC    (SYS_CLK / 21600L)
volatile WORD    spinx;
void    spin(WORD ms)
{
        WORD    i;
        while (ms--)
                for (i = 0; i < SPINC; i++) spinx = i*i;
}

//      END
```

```
#       GCC MAKEFILE:
#
#       GMAKE file for the "oliver" code running on the ATmega32U4.
#
CC      = avr-gcc.exe
MCU     = atmega32u4

#       These are common to compile, link and assembly rules
#       The 'no-builtin' opt keeps gcc from assuming defs for putchar() and
#       others
#
COMMON  = -mmcu=$(MCU) -fno-builtin

CF      = $(COMMON)
CF      += -Wall -gdwarf-2
##CF    += -Wall -gdwarf-2 -O0

AF      = $(COMMON)
AF      += $(CF)
AF      += -x assembler-with-cpp -Wa,-gdwarf2

LF      = $(COMMON)
LF      += -Wl,-Map=$(TMP)$(APP).map

#       weird intel flags
#
HEX_FLASH_FLAGS  = -R .eeprom

HEX_EEPROM_FLAGS = -j .eeprom
HEX_EEPROM_FLAGS += --set-section-flags=.eeprom="alloc,load"
HEX_EEPROM_FLAGS += --change-section-lma .eeprom=0 --no-change-warnings

#       -------------------------------------------------------------------
#       MAKE DIRECTIVES
#
#       The "target directory" TMP is standard for the intermediate files
#       (.obj) and the final product.
#       -------------------------------------------------------------------
TMP     = ./tmp/

APP     = oliver
ELF     = $(TMP)$(APP).elf
OBJS    = $(TMP)oliver.o

all:    $(TMP) $(ELF) $(TMP)$(APP).hex $(TMP)$(APP).eep size

$(TMP):
        rm -rf .\tmp
        mkdir .\tmp

$(TMP)oliver.o: oliver.c
        $(CC) $(INCLUDES) $(CF) -O1 -c  $< -o $*.o

oliver.asm: oliver.c
        $(CC) $(INCLUDES) $(CF) -O1 -S  $< -o oliver.asm
```

```
#        Linker  ---------------------------------------------------------
$(ELF): $(OBJS)
         $(CC) $(LF) $(OBJS) $(LINKONLYOBJS) $(LIBDIRS) $(LIBS) -o $(ELF)

%.hex: $(ELF)
         avr-objcopy -O ihex $(HEX_FLASH_FLAGS)  $< $@

%.eep:   $(ELF)
         -avr-objcopy $(HEX_EEPROM_FLAGS) -O ihex $< $@ || exit 0

%.lss:   $(ELF)
         avr-objdump -h -S $< > $@

size:    $(ELF)
         @echo
         @avr-size -C --mcu=${MCU} $(ELF)

#        Misc  -----------------------------------------------------------
clean:
         -rm -rf $(OBJS) $(TMP)$(APP).elf ./dep/* $(TMP)$(APP).hex $(TMP)$(APP).eep
         $(TMP)$(APP).map $(TMP)$(APP).d

#        end
```

Information furnished by Linear Technology Corporation is believed to be accurate and reliable. However, no responsibility is assumed for its use. Linear Technology Corporation makes no representation that the interconnection of its circuits as described herein will not infringe on existing patent rights.

I taste good, go well with other foods, am healthy to eat and measure temperature to .1°F resolution.
Reese
CANNONBALL
Olives
Stuffed with Minced Pimiento on table server
PRODUCT OF SPAIN
NET WT. 6 OZ. (170 g)
'10

Linear Technology Corporation
1630 McCarthy Blvd., Milpitas, CA 95035-7417
(408) 432-1900 • FAX: (408) 434-0507 • www.linear.com

LT 0411 • PRINTED IN USA

© LINEAR TECHNOLOGY CORPORATION 2011