Vishay's TSSP4056 Sensor for Fast Proximity Sensing

By John Fisher

In our previous application note, "Vishay's TSSP-AGC P Sensor Series for Proximity Sensing," it was shown how a TSSP4P38 sensor could be used with minimal external hardware to detect not only the presence of an object with infrared light, but whether that object was moving nearer or further away from the sensor. We define proximity sensing, as opposed to presence sensing, as detecting the movement of an object away or towards the sensor (or the movement of the sensor away or towards the object). The TSSP4P38 has one drawback that may limit its use in some proximity sensing applications: it is quite slow. The long reaction and recovery times of the AGC limit the usefulness of the TSSP4P38 to only about two readings per second.

In this application note we introduce a new sensor, the TSSP4056, and an example software program to accomplish the same task in approximately 5 ms, giving us about 190 readings per second. To illustrate the difference in speed between these two techniques, the TSSP4P38 can resolve the position of a person walking at an average speed (generally about 1.4 m/s) to an accuracy of within about 70 cm, while the TSSP4056 can resolve the position of the same person to within about 3/4 centimeter. Further, the TSSP4056 can accomplish this feat with no increase in hardware.

THE TSSP4056

The TSSP4056 is of the constant gain or AGC "0" type, and contains a band-pass filter tuned so that the sensor reaches its highest sensitivity for signals at 56 kHz. Like any receiver based on remote control technology, the TSSP4056 can also receive signals at nearby frequencies with a lower gain. The constant gain AGC0 refers to the sensor's ability to change its amplification characteristics in the presence of modulated infrared noise coming from CFL lights, for example. It is generally desirable for remote control receivers to automatically reduce their gain under such conditions to avoid generating spurious pulses.

As will be shown, the proximity sensor presented here crucially depends on its gain not changing in response to ambient lighting, nor to the very long burst signals we will be emitting, which have very similar characteristics to noise. We will use the gain vs. frequency response of the sensor to determine the distance to an object, and any change in gain due to the AGC being triggered from ambient lighting or our emitter signal would erroneously appear as if the object were moving.

THE FAST PROXIMITY SENSOR

The operating principle for a fast proximity sensor is quite simple. Let us look first at the general-purpose case, then at an alternative solution. The infrared emitter transmits a short burst of about 25 cycles at 56 kHz at its highest output power (a burst is simply a pulse that is modulated with a carrier frequency). If the sensor detects a reflection off an object, an active low pulse at the sensor output with the modulation removed will validate this. We then know an object is in range, but know nothing of its relative distance. The emitter then transmits a slightly less powerful burst and checks for validation by the sensor. If the pulse is again detected, we continue this process of lowering the emitter burst power until no reflection is seen. The emitter power at this threshold level is then related to the distance of the object from the sensor.

A conceptual block diagram to perform this process is shown below. The microcontroller provides a digital value to the DAC, which converts this to an analog voltage. The oscillator switches the positive input of the op-amp between the analog voltage from the DAC and ground to create a burst with a defined peak voltage. The op-amp and transistor reproduce the burst across the resistor, which converts the voltage burst to a current burst to drive the emitter. Note that emitter power is linearly related to the emitter forward current, which is defined by the voltage across the resistor.

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

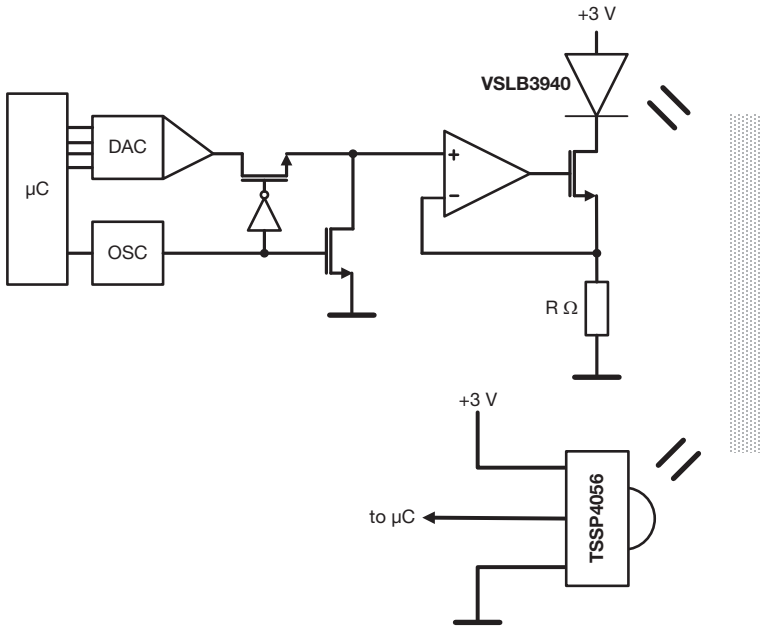

Fig. 1

While this implementation may work quite well, its main drawback is cost, especially for precision components like the DAC. Our goal for this project is to perform fast proximity sensing without adding components to our TSSP4P38 sensor solution: an emitter plus resistor, a sensor, a plastic holder, and an Arduino Nano.

AN ALTERNATIVE CONCEPT

As in our previous application note on the TSSP4P38, the burst signal for driving the emitter will be generated using one of the Nano's three on-board timers, and another on-board timer will be used to capture and validate the sensor's output. While the Nano has no provision for generating different voltage levels from its timers, it can easily generate different frequencies under software control. If we note from the TSSP4056 datasheet that the sensitivity of the sensor follows a typical band-pass response (see figure below), then an alternative concept for finding the threshold sensitivity of an object at an unknown distance begins to emerge.

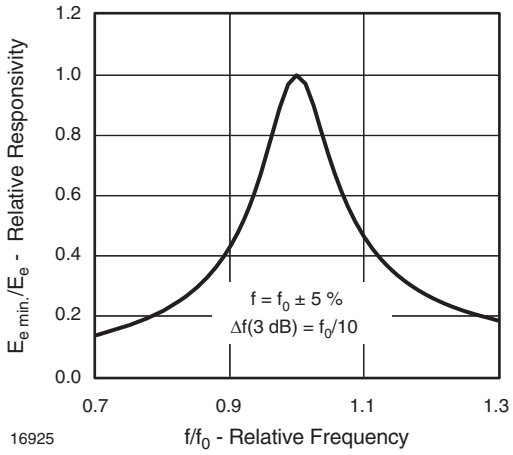

Fig. 2

The infrared emitter again transmits a short burst of about 25 cycles at 56 kHz, which corresponds to the frequency where the

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

sensor is most sensitive and has its highest range. If the sensor detects a reflection off an object, again validated by an active low pulse, we know an object is in range. The emitter then transmits a burst at a slightly lower frequency and checks for validation by the sensor. The sensor will be less sensitive to this pulse, and its range will be lower. If the pulse is again detected, we continue this process of lowering the burst frequency until no reflection is seen. The burst frequency at this threshold level is then related to the distance of the object from the sensor. In our example software, we implement this process using successive approximation to maximize the speed. The threshold distance is found after eight successive approximations, giving us a reading in approximately 5 ms for 25 cycle bursts. The exact time for a reading is dependent on the distance of the object. Far objects are detected faster than near objects because of the higher frequencies used to detect them.

In order to check the feasibility of this concept, pulses at different frequencies were generated and the threshold detection distance from the sensor to a sheet of white paper was measured, as shown in the chart below. While the relationship is not exactly linear, it is monotonic below the peak sensitivity, which is important for finding a single solution via successive approximation. To avoid non-monotonicity around the peak sensitivity, the maximum burst frequency used in the software was backed off slightly to 54 kHz. The distance vs. frequency characteristic is in any case linear enough to allow use of this technique for a large range of applications, from robot navigational avoidance to gesture control via hand-sensor distance.

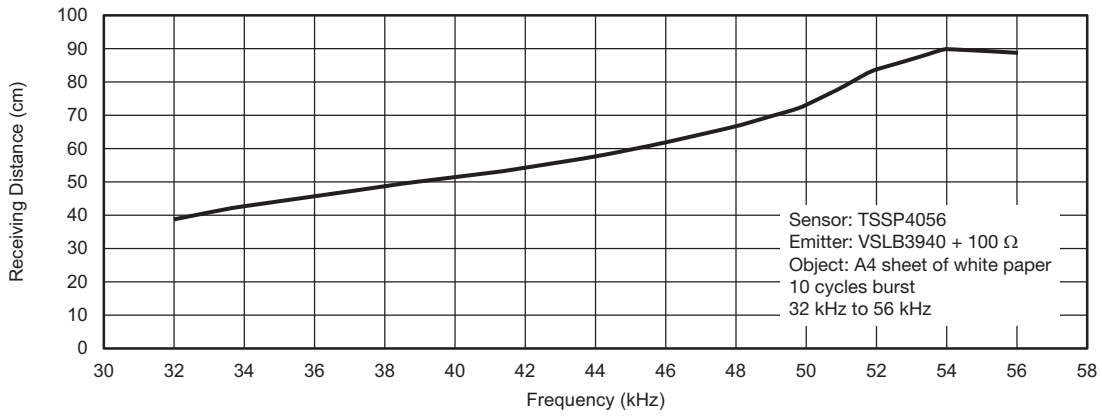

Fig. 3

This graph was generated using a 100 Ω resistor, which was selected in order to not exceed the Nano's maximum I/O current of 40 mA. The VSLB3940 can be driven with much higher currents if larger detection ranges are required. In that case, an external transistor driver should be inserted between the Nano and the emitter to boost the driving current. In case closer ranges are required, a larger resistor value should be selected to reduce the emitter current.

Some applications may require a higher dynamic range than is obtainable using the gain-frequency characteristics of the sensor alone. This would be the case if both long and short ranges need to be detected; for example, from several cm to one meter in the graph above.

It may seem obvious that generating even lower burst frequencies should result in additional dynamic range. Unfortunately, this technique should not be used. At frequencies lower than 32 kHz, the TSSP4056 passes through a non-linear region in its frequency / sensitivity curve. This has to do with the shape of the band-pass filter characteristic, where the gain actually begins to increase again before resuming its downward trend.

One possibility for increasing the dynamic range is to auto-range the emitter current by switching in one of several higher-value resistors in the emitter circuit. A higher-value resistor results in a lower emitter current, and consequently a lower range. The cost of this modification is the cost of the resistors themselves and two or three unused I/O pins on the Nano, as shown below. The "ground" side of the unused resistor(s) should be tri-stated, and the output to the active resistor should be output low.

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

The logic changes as follows. If the successive approximation reaches its minimum sensitivity with the lowest value resistor, and the sensor still shows a reflection, a higher-value resistor would be switched in and the process repeated. The main challenge in this technique is choosing a resistor-frequency combination where the low-value resistor plus the minimum frequency gives approximately the same range as the high-value resistor plus the maximum frequency.

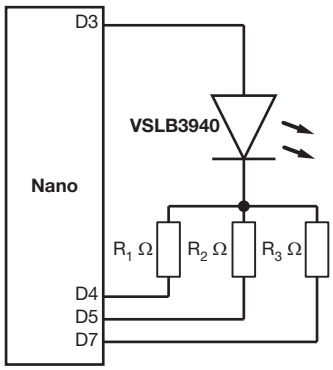

Fig. 4

THE HOUSING

An essential part of a reflective sensor is a suitable housing in which to mount the optical components. The housing must not only provide mechanical support, it must guarantee complete optical isolation between the emitter and the receiver. No path may exist between the emitter and the sensor other than the reflection path from the detected object. The gain of the sensor is very high and even small amounts of stray light, e.g. reflected off stray wiring, can activate the sensor and render it blind to reflections from the far field. Care must also be taken to avoid using a common transparent window between the emitter and sensor. This window may act as a light guide between the two components.

Shown below is the TSSP-HA sensor housing, available through our distribution partners, which allows rapid prototyping in standard breadboards.

Fig. 5

To the left, a TSSP4056 and VSLB3940 in a plastic housing provide mechanical support and optical isolation.

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

ARDUINO NANO BOARD

A fast, easy-to-use solution for generating the signals and logic for the VSLB3940- and TSSP4056-based sensor is a low-cost development board, such as the open-source Arduino Nano. Please see our TSSP4P38 application note for details on the Nano. The electrical connections are simple, as shown below:

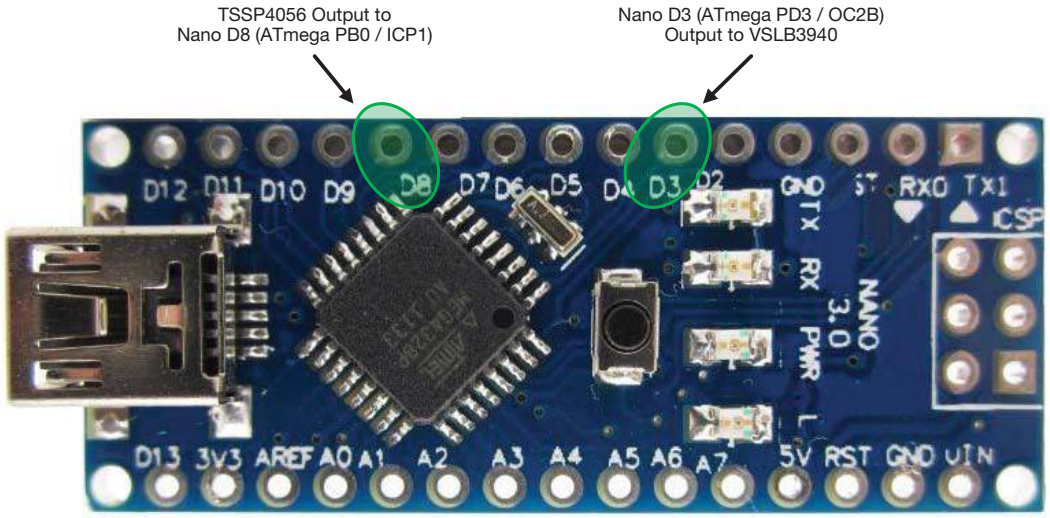

Fig. 6

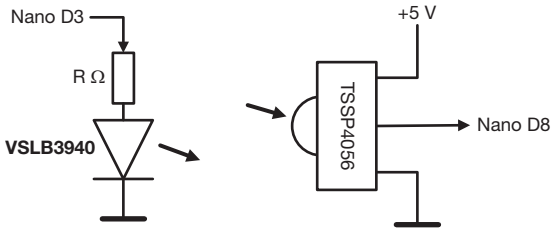

Fig. 7

APPLICATION NOTE

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

THE EXAMPLE SOFTWARE PROGRAM

Create a folder called TSSP4056_FastProximity_Sensor in the location where your installation sets up the Sketchbook, and unpack the example program in this folder. You should now be able to find and open the file from the IDE menu item File → Sketchbook. After compiling and uploading the program to the Nano, open the Serial Monitor and make sure the baud rate is set to 57 600. A much higher baud rate is used in this project than in the TSSP4P38 example. This is needed to speed up the serial interface that limits the speed of the sensor, essentially by waiting to transfer the readings to the serial monitor. Also, the Arduino serial library routines are not used for the same reason. Instead, the UART registers are written to directly.

A visual bar graph is continually drawn and scrolled in the monitor window with the X-axis representing the distance of an object from the sensor. The scrolling of this output is very fast to the human eye, giving an impressive demonstration of the sensor speed.

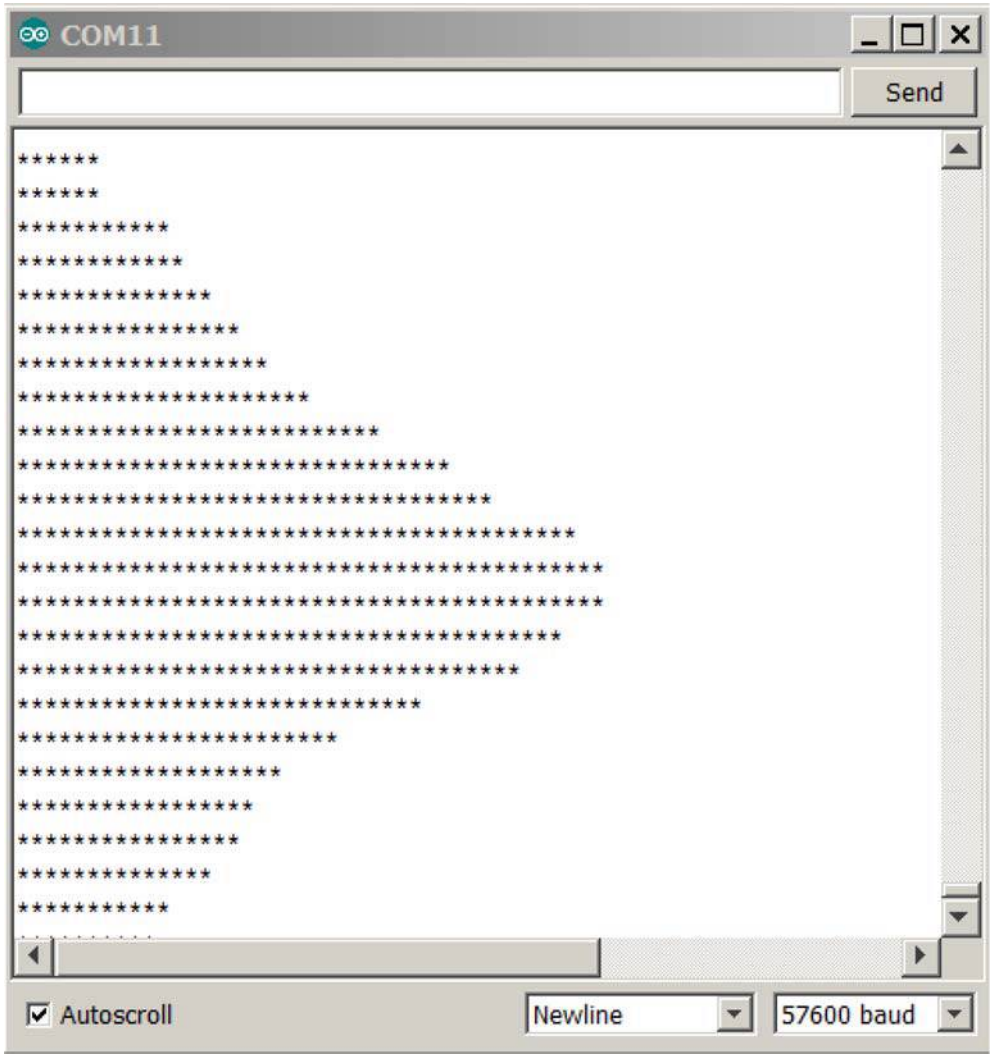

Fig. 8

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

TIMERS, TIMERS, TIMERS - THE EMITTER

The emitter will be driven by the OC2B output of the 8-bit Timer2 used in mode 5, which is defined as a "PWM phase correct mode" with the TOP value determined by the value of the output compare register OCR2A. In mode 5, Timer2 counts up from zero to the value of OCR2A and back down again, generating an overflow interrupt (if enabled) each time the counter passes zero. Timer2 gets its clock from the system with no pre-scale. Each journey from zero to OCR2A and back corresponds to one cycle of our burst signal. Another output compare register, OCR2B, determines the value at which the output OC2B switches. We use the setting: clear OC2B on match counting up and set OC2B on match counting down. The value at which this occurs (the value of OCR2B) is recalculated each time we change the frequency, or OCR2B = OCR2A/2, giving us a 50 % duty cycle at all times.

It should be clear by now that the burst frequency is set by the value in OCR2A. The time to count from zero to the value in OCR2A is equal to the time for each count of the timer (1/16 MHz) times OCR2A. This is equal to half the period (T/2) of the desired frequency, or $1/2f$. So $1/2f$ = OCR2A/16 MHz, or rearranging f = 16 MHz/(2 x OCR2A), or conversely OCR2A = 16 MHz/$2f$. Really, we are only concerned with knowing the two endpoints, 32 kHz (OCR2A = 250) and 54 kHz (OCR2A = 148), since the successive approximation routine just treats the range between these endpoints as a series of numbers and does not concern real-world frequencies.

The interrupt service routine (ISR) for Timer2 uses a static variable (a local variable that persists after the routine exits) that is incremented each time the ISR is called. This is our carrier cycle counter. When the cycle count reaches the burst length, the cycle counter is reset and Timer2 is stopped by setting the clock select bits to zero.

TIMERS, TIMERS, TIMERS - THE SENSOR

Before discussing the sensor timer, let us take a look at a very important parameter on the TSSP4056 datasheet: t_{po}. You can see in the figure below an optical burst of length t_{pi}. That is the emitter signal we generated in the last section. You can also see the sensor's response to t_{pi} in the lower half of the figure, a pulse of length t_{po}.

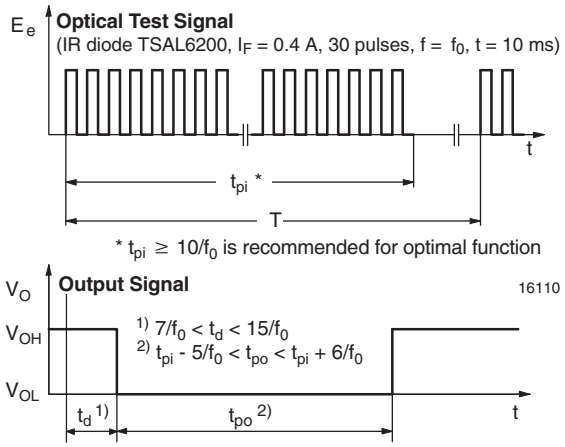

Fig. 9

Generally, the length of the output pulse t_{po} is strongly dependent on the strength, or power, of the received signal, and by implication, also its deviation from the band-pass center frequency. What this figure says is that the length of the output pulse is guaranteed to be greater than the length of the input burst less five cycles of the carrier frequency, and less than the length of the input burst plus six cycles of the carrier frequency $\left(t_{pi} - \frac{5}{f_0} < t_{po} < t_{pi} + \frac{6}{f_0}\right)$.

In other words, the sensor has some tolerance in the way it reproduces the input burst. One of the primary reasons for using Timer1 to measure the length of the sensor's pulse instead of just using a pin change interrupt is to measure and validate the result. If the t_{po} we measure is outside the tolerances, this will be regarded as an invalid pulse and ignored. A strict interpretation of this datasheet parameter is, however, counterproductive. We are using valid signal frequencies quite far from the band-pass center frequency. This use results in an even wider tolerance in the direction of shorter pulses than is given in the datasheet.

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

The example software achieves good results using $t_{pi} = \frac{25}{f_0}$ for its emitter pulse and allows the sensor output pulse to be up to 10 cycles less, or $t_{pi} - \frac{15}{f_0} < t_{po}$.

The sensor output will be measured with the noise-cancelling input of the 16-bit Timer1 via edge-triggered interrupts. Timer1 is set up to use the system clock with a pre-scale of 64, which provides a higher resolution than one period of the carrier signal. The SendBurst function that handles the process of sending a burst and measuring the reflection sets Timer1 for a falling edge interrupt. The ISR of Timer1 first checks whether interrupts are set for rising or falling edge. If falling edge, it concludes that a pulse has just commenced and resets the value of Timer1 to zero. It also sets interrupts for rising edge, then exits. If rising edge interrupts are active on entry to the ISR, the routine concludes that a pulse has just ended. It stores the value of Timer1 in a volatile variable, sets interrupts for falling edge, and exits.

The SendBurst function accepts parameters for the half period of the carrier frequency (essentially the OCR2A value in the discussion above), the burst length in cycles (default 25), and the duty cycle (default 50 %). From the half period, it calculates the upper and lower pulse tolerances, PulsTolHi and PulsTolLo, used to validate the length of the reflected pulse. The function returns -1 to signal that a burst was sent but no pulse was received, 2 to signal that a pulse was received but was out of specification, or the value of Timer1 for a valid pulse.

THE MAIN ROUTINE - SUCCESSIVE APPROXIMATION

The code for the successive approximation routine is surprisingly concise, but not particularly easy to understand at a glance. Instead of working with frequencies, we are using half periods, or the values that are loaded into Timer2's OCR2A. Each pass of the loop function gets one sensor reading and passes the value to the serial monitor. At the start of a pass, three variables are initialized: HalfPeriodHi is set to the longest period (lowest frequency) for our sensor; HalfPeriodLo to the shortest period (highest frequency); and HalfPeriodTry to the period our routine will try in its next burst. The initial value of HalfPeriodTry is equal to HalfPeriodLo, or the most sensitive frequency.

SendBurst is continually called until a valid pulse is received. Once this occurs, the sensitivity must now be reduced to half of its current value, so HalfPeriodTry is set to the midpoint between HalfPeriodLo and HalfPeriodHi. If SendBurst now gives us a valid pulse, we set HalfPeriodLo to the value in HalfPeriodTry, resulting in a further reduction in sensitivity. If no valid pulse is received, we set HalfPeriodHi to the value in HalfPeriodTry, resulting in an increase in sensitivity. Now the new value of HalfPeriodTry is calculated, and a SendBurst is called again. After eight passes, the approximation has completed. The flow chart below depicts the logic. Manually working through the process can provide a clearer understanding and will convince you that the logic works.

Vishay's TSSP4056 Sensor for Fast Proximity Sensing

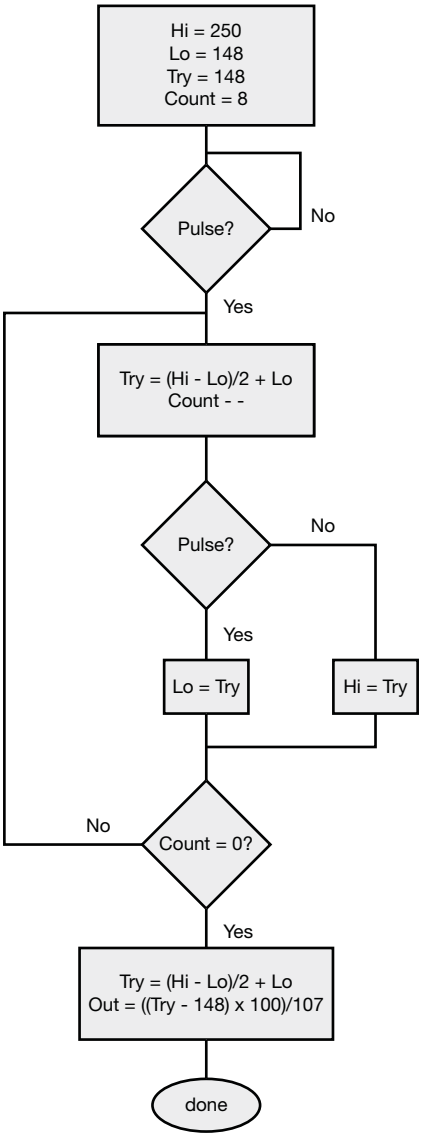

Fig. 10

APPLICATION NOTE