Implementation of high-resolution time-to-digital converter in 8-bit microcontrollers

Lars E. Bengtsson
Department of Physics, University of Gothenburg, SE-412 96 Gothenburg, Sweden

(Received 6 February 2012; accepted 17 March 2012; published online 5 April 2012)

This paper will demonstrate how a time-to-digital converter (TDC) with sub-nanosecond resolution can be implemented into an 8-bit microcontroller using so called "direct" methods. This means that a TDC is created using only five bidirectional digital input–output-pins of a microcontroller and a few passive components (two resistors, a capacitor, and a diode). We will demonstrate how a TDC for the range 1–10 μs is implemented with 0.17 ns resolution. This work will also show how to linearize the output by combining look-up tables and interpolation. © *2012 American Institute of Physics*. [http://dx.doi.org/10.1063/1.3700192]

I. INTRODUCTION

A time-to-digital converter (TDC) quantizes the time between a start and a stop signal and they are either analog or counter-based.[1–3] Analog TDCs use the start and stop signals to generate a pulse with a length corresponding to the start and stop times, this pulse is then converted into an analog voltage by an integrator, and then an analog-to-digital converter (ADC) digitizes this voltage, see Figure 1.

The aim of this work was to design a high-resolution (sub-nano) TDC using a simple inexpensive 8-bit microcontroller only. For several reasons, the analog TDC technique in Figure 1 was discarded. First, the ADC in popular microcontrollers such as Microchip's PIC18 family of controllers has a resolution of only 10 bits.[4] Second, we would like our design to work also for inherently digital systems like field programmable gate arrays (FPGAs) and complex programmable logic devices.

Hence, we need to focus on counter-based TDCs. A "basic" counter-based TDC simply counts the pulses from a reference clock during the start and stop interval, see Figure 2.

Since the start and stop signals are asynchronous with the reference clock, there will be an inherent ± 1 count uncertainty, and hence the time resolution is limited by the reference clock's speed.[1] However, increasing the clock frequency raises two issues; first the power consumption increases and second there is a limit to the maximum oscillator frequency that can be implemented in CMOS.[1] (If the start and stop signals are asynchronous to the reference clock, the resolution can be increased by averaging.)

An 8-bit PIC-controller from Microchip running with a 20 MHz crystal will internally run at 5 MHz.[4] Using its embedded 16-bit timers for time interval measurements, using the basic counting technique, would render a maximum time resolution of ± 200 ns. In order to achieve sub-nano resolution (without averaging) we need more advanced methods.

There are a few commonly used principles that can be implemented in order to increase the resolution of counter-based TDCs; tapped delay-lines, vernier clocks, or time stretching. Tapped delay-lines use digital buffers as delay elements where each buffer's output is connected to the input of an edge-triggered flip-flop. The flip-flops are latched by the stop signal arriving some time later, producing a "temperature"-coded time interval value on the flip-flops' output,[1,5] see Figure 3.

This is a very fast, high-resolution TDC and sometimes is referred to as a "flash TDC" (representing the TDC correspondence to flash ADCs[6,7]). Even though the tapped delay-line solution is fast and all-digital, it is not suitable for implementation in "fixed-hardware" targets such as microcontrollers; there are simply no buffers and flip-flops available on-chip.

In the vernier principle, high-resolution TDCs are implemented by employing two oscillators with slightly different frequencies $f_1 = 1/T_1$ and $f_2 = 1/T_2$, respectively.[1,2,8,9] The start and stop signals trigger one oscillator each, see Figure 4.

Oscillator 1, with frequency f_1 ($<f_2$), starts on the positive edge of the start signal. The second oscillator, with frequency f_2, is triggered on the positive edge of the stop signal. Since $f_2 > f_1$, the pulses from the f_2-oscillator will eventually catch up with the pulses from the f_1-oscillator. When this happens, both counters are stopped (and notice that at this point both oscillators have generated exactly the same number of pulse, i.e., $N_1 = N_2 = N$). From Figure 4 it is obvious that

$$\tau = N_1 T_1 - N_2 T_2 = N\,(T_1 - T_2) = N \times \Delta T. \qquad (1)$$

The time resolution depends on the time difference ΔT of the clocks' cycle periods. This idea could be implemented in a microcontroller as long as it has two separate timers/counters that can be clocked independently. However, it would be hard to reach the theoretical time resolution ΔT if we implemented this method in a typical microcontroller from Atmel/Microchip/Freescale. In order to achieve the theoretical time resolution ΔT, we would have to be able to compare the timer registers in hardware; comparing timer registers in a polling firmware algorithm is too time consuming and we

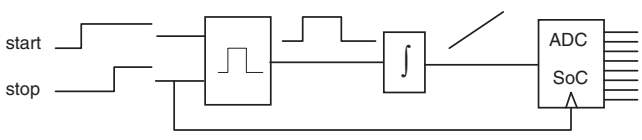

FIG. 1. Schematic principle of analog TDC.

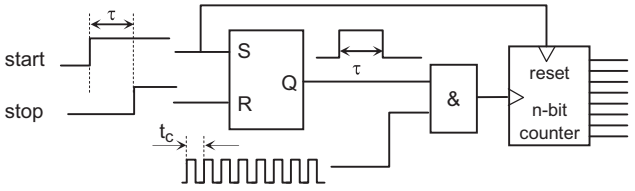

FIG. 2. Schematic principle of basic counting TDC.

would most likely miss the moment of timers' coincidence. That method was indeed examined in this work and when we used an 8-bit PIC18F controller from Microchip, the minimum uncertainty in the moment of coincidence detection was ±30 counts (for $\Delta T = 4.6$ ns). Neither Microchip nor Atmel has any 8-bit controllers that can compare two 16-bit timer registers in hardware. (They can compare 16-bit registers in hardware, just not two running timer registers in real-time.)

In order to implement a high-resolution TDC in an 8-bit controller, we are left with the technique of time stretching. In the basic counter-based TDC, the time resolution is $\pm t_c$, t_c being the oscillator clock cycle period in Figure 2; if we count N pulses, our estimate of the time interval τ is

$$\hat{\tau} = (N \pm 1) \times t_c. \tag{2}$$

The resolution is t_c and the uncertainty is $\pm t_c$. Suppose we stretch the time interval by a factor of k, see Figure 5.

If we measure the stretched time interval τ' with the same instrument as in Figure 2, we would get N' pulses:

$$\hat{\tau}' = \left(N' \pm 1\right) \times t_c = N' t_c \pm t_c, \tag{3}$$

but since $\tau = \tau'/k$, our estimate of τ is

$$\hat{\tau} = \frac{1}{k}\hat{\tau}' = N' \frac{t_c}{k} \pm \frac{t_c}{k}. \tag{4}$$

Hence, if the time is stretched by a factor of k, both the resolution and the uncertainty are improved by a factor of k.

There are several ways to stretch a time interval and Figure 6 illustrates the basic principle; charging and discharging of a capacitor.[10,11]

In Figure 6 it is assumed that $i_1 >> i_2$. The capacitor is charged during τ by a constant current $i_1 - i_2$ and then discharged by i_2 only. This will produce a prolonged pulse on the comparator's output that will be $k = i_1/i_2$ times longer than τ.[10]

This paper will describe how time stretching can be implemented with a simple 8-bit microcontroller and a few passive components only. The rest of this paper is organized as follows; in Sec. II we will present a new time stretching idea that can be implemented in a microcontroller with only a few passive external components. This idea is based on the "direct sensor-to-controller"[12–18] and "direct analog-to-controller"[19] techniques and therefore these techniques are introduced first. In Sec. III we analyze the proposed circuit and present some theoretical expressions and design rules. In Sec. IV we present some experimental results and in Sec. V we briefly describe how a look-up table (LUT) is implemented in order to produce a linear output response. Section VI summarizes the reported work and results and draws some final conclusions.

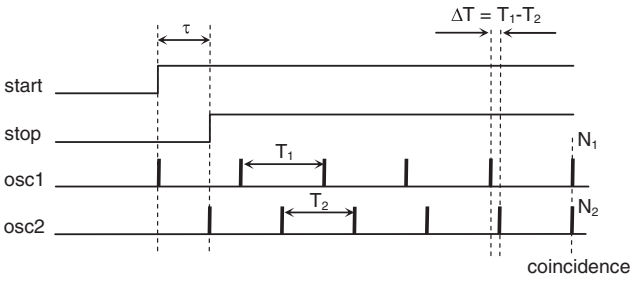

FIG. 4. The vernier principle.

II. THE "DIRECT" MICROCONTROLLER TECHNIQUE

In the late 1990s, several reports were published concerning the technique of "direct" interfacing of passive sensors to microcontrollers.[12,13] With this technique, passive sensors such as resistive thermal devices (RTDs) and capacitive humidity sensors can be interfaced to digital embedded systems that do not have an embedded ADC. Figure 7 illustrates the fundamental idea.

The technique is based on the fact that digital input–output (I/O)-pins on a typical microcontroller (or FPGA) are bidirectional and reconfigurable in firmware; as inputs they represent high-impedance (high-Z) loads and as outputs they represent low-impedance loads that can be either sourcing (set high) or sinking (set low). In Figure 7, R_S represents the RTD and R_C represents a calibration resistor. During the first stage, I/O-pins 1 and 2 are configured as inputs (high-Z) and I/O-pin 3 is configured as output and set high. I/O-pin 3 will charge the capacitor to V_{OH} (=the output high voltage of I/O-pin 3). During the second stage, the capacitor is discharged through R_S by reconfiguring I/O-pin 2 to out-

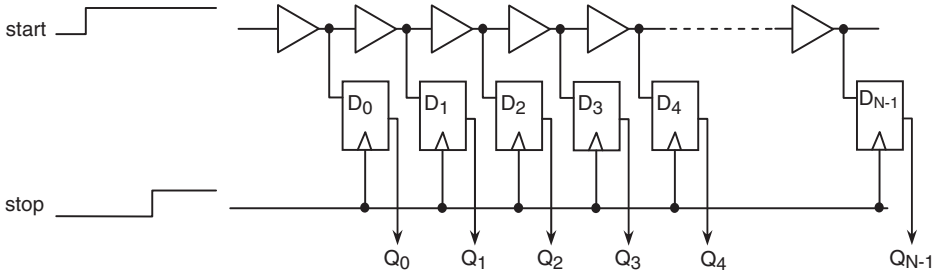

FIG. 3. A basic delay-line for generation of clock cycle subdivisions.

FIG. 5. Time stretching.

put (set low) and I/O-pin 3 is reconfigured to input. I/O-pin 3 is polled in firmware until the capacitor has been discharged to V_{IL} (=input logic low threshold of I/O-pin 3). An internal timer measures this time (producing an integer N_S). Next, the charging/discharging procedure is repeated, but now the capacitor is discharged through R_C, and the internal timer produces an integer N_C. From these two integers, the unknown resistance R_S may be estimated:[12]

$$\hat{R}_S = \frac{N_S}{N_C} \times R_C. \tag{5}$$

Richey[20] suggested a similar method for direct interfacing of capacitive sensors and Reverter and Casas[21] presented a similar solution for differential capacitive sensors. In 2001, Custodio *et al.*[14] presented a direct approach to the interfacing of bridge sensors and in 2012, Bengtsson[19] reported a work on how to interface analog voltage output sensors "directly" to digital embedded systems.

This work now suggests a "direct" microcontroller solution for TDC implementation by time stretching. The idea is illustrated in Figure 8.

At first, all I/O-pins are configured as inputs (high-Z), except for I/O-pin 4 that is configured as output and set low. When the controller is interrupted (on the interrupt pin) by the positive pulse edge, the capacitor is charged via R_1 during the time interval τ; C will be charged to a voltage proportional to the time interval τ. I/O-pin 1 is polled in order to detect the negative edge (=the stop signal). When I/O-pin 1 senses the negative pulse edge, I/O-pin 2 is reconfigured to output and set low and the capacitor is discharged through R_2. (Notice how the diode directs all discharging current into R_2.) At the same time as I/O-pin 1 detects the negative pulse edge, I/O-pin 4 is also set high. Assuming that the capacitor was charged to a voltage $>V_{IH}$ (input voltage logic high for I/O-pin 3), polling I/O-pin 3 will detect the time it takes for the capacitor to discharge to V_{IL} (input voltage logic low for I/O-pin 3). When V_{IL} is detected, I/O-pin 4 is reset. The pulse width of I/O-pin 4 will be a stretched version of the input pulse if $R_2 \gg R_1$; Figure 8 is indeed a pulse stretcher and may be used for implementing high resolution TDCs in 8-bit microcontrollers (and of course 16/32-bit controllers too). Figure 9 illustrates the firmware flowchart.

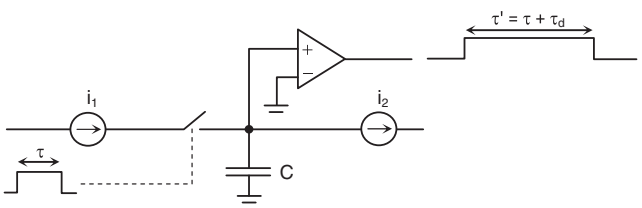

FIG. 6. Time stretching by Nutt.[10]

III. SYSTEM ANALYSIS

Figure 10 illustrates the equivalent circuit during the charging.

If U_{in} is the amplitude of the input pulse and U_d is the voltage drop across the diode, the capacitor C is charged by a voltage $U_0 = U_{in} - U_d$ during the time interval τ. After time τ, C is charged to U_τ:

$$U_\tau = U_0 \times (1 - \mathrm{e}^{-\tau/R_1C}). \tag{6}$$

For this design to work, it is important that the time constant R_1C is chosen such that $U_\tau > V_{IH}$; U_τ must be large enough so that it exceeds the input logic high threshold of I/O-pin 3. This will set a limit to the minimum pulse width we can analyze. On the other hand, the capacitor should not be saturated ($U_\tau < U_0$) because if it is saturated, τ cannot be uniquely determined. For the design to work reliably, we require that the capacitor is never charged to more than ~99% of U_0. That gives us the following conditions:

$$V_{IH} < U_\tau < 0.99U_0. \tag{7}$$

Inserting (6) into (7) gives us

$$V_{IH} < U_0 \times \left(1 - \mathrm{e}^{-\tau/R_1C}\right) < 0.99U_0 \tag{8}$$

and from Eq. (8) we get either a condition for R_1C depending on the desired range of τ or an expected range of τ for any given time constant R_1C:

$$-R_1C \times \ln\left\{1 - \frac{V_{IH}}{U_0}\right\} < \tau < 4.606 \times R_1C, \tag{9}$$

$$-\frac{\tau}{\ln\left\{1 - \frac{V_{IH}}{U_0}\right\}} < R_1C < 0.2171 \times \tau, \tag{10}$$

where τ must be in the range given by Eq. (9) for C to be charged to a voltage in the recommended range V_{IH}–$0.99U_0$.

During phase 2, the capacitor is discharged through R_2. The equivalent circuit is illustrated in Figure 11.

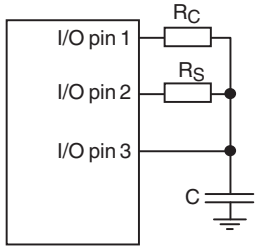

FIG. 7. The one-point calibration "direct" technique.[12]

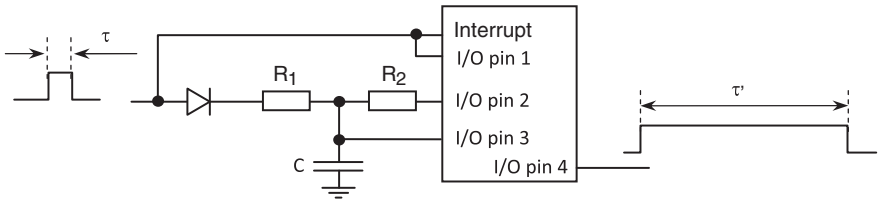

FIG. 8. A "direct" microcontroller TDC.

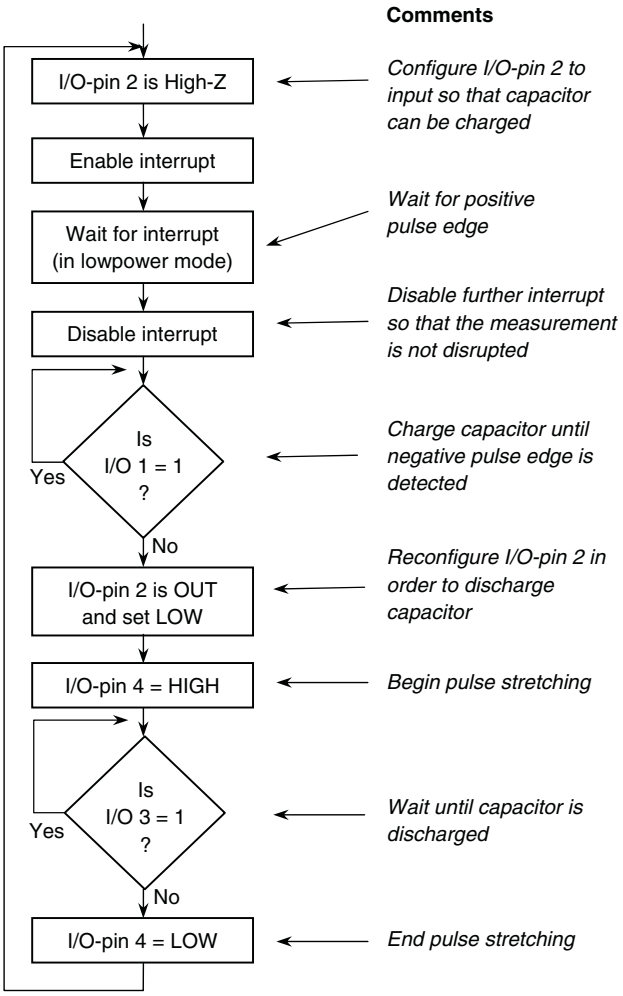

FIG. 9. Firmware flowchart.

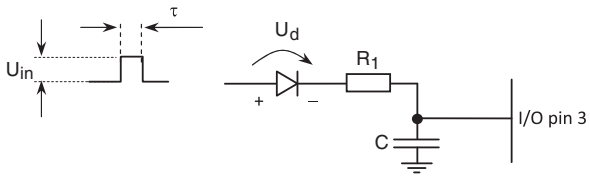

FIG. 10. Equivalent system circuit during charging.

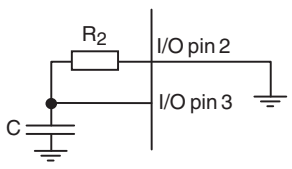

FIG. 11. Equivalent circuit during discharging.

If the capacitor voltage $U_\tau > V_{IH}$ at $\tau = 0$, it will take some time t_d to discharge it to V_{IL}:

$$U_d(t) = U_\tau \times \mathrm{e}^{-t/R_2C} \Rightarrow t_d = -R_2C \times \ln\left\{\frac{V_{IL}}{U_\tau}\right\},$$
$$t_d = -R_2C \times \ln\left\{\frac{V_{IL}}{U_0(1-\mathrm{e}^{-\tau/R_1C})}\right\}, \quad (11)$$

where t_d is the time during which I/O-pin 4 is set high. Dividing by the input pulse width gives us the time expansion factor k:

$$k = \frac{t_d}{\tau} = \frac{\tau'}{\tau} = -\frac{R_2C}{\tau} \times \ln\left\{\frac{V_{IL}}{U_0(1-\mathrm{e}^{-\tau/R_1C})}\right\}. \quad (12)$$

IV. EXPERIMENTAL RESULTS

The time stretching technique in Figure 8 was implemented in a PIC18F4580 microcontroller from Microchip.[4] The V_{IL} and V_{IH} parameters were measured, as described by Reverter *et al.*,[18] to 1.23825 V and 1.25384 V, respectively. To generate the input pulse, we used a HP8013B pulse generator and U_0 was measured to 3.16 V using a Tektronix oscilloscope TDS2012B. In order to demonstrate the idea and to verify theoretical expressions, we first chose the time constants to get a range of approximately 10–100 μs. R_1 and R_2 were measured to 99.71 Ω and 8.078 kΩ, respectively (using a Phillips PM2534 DMM and the 4-wire method). C was measured to 207 nF (measured with a HP4216A LCR meter). According to expression (9), this suggests a range for τ as follows:

$$10.43 \times 10^{-6}\mathrm{s} < \tau < 95.05 \times 10^{-6}\mathrm{s}.$$

In Figure 12 we have plotted experimentally registered data; measured stretch factor versus input pulse width. For comparison we have also plotted expression (12) in the same diagram.

As we can see in Figure 12, observed data deviates some from the theoretical predictions. The theoretical predictions represent expression (12) and we examined that for some small disturbances in C and U_0, and it turns out to be very sensitive to these variations. A possible explanation for the deviations between observed data and theoretical predictions is first that the total capacitance is not just the capacitor C; the equivalent circuit diagram should probably also include the diode's capacitance. The diode capacitance is particularly important at high frequencies and when the diode operates in the reverse direction[22,23] (as it does during discharging). Another contribution to the deviations from the theoretical prediction is that the diode distorted the input pulse shape, so that it was

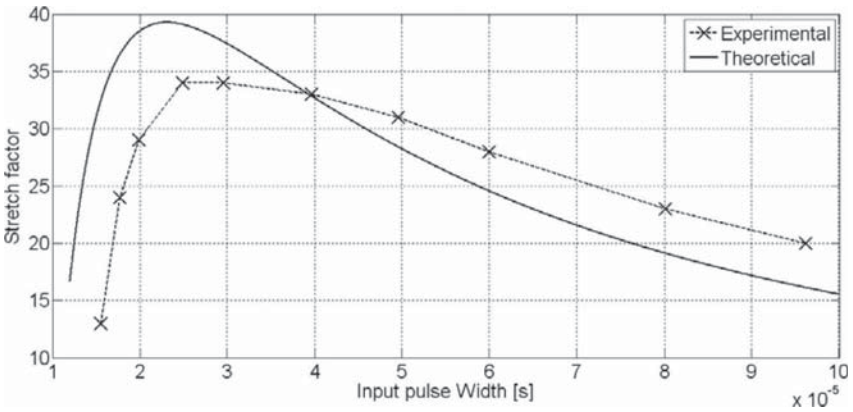

FIG. 12. Observed stretch factor compared with theoretical stretch factor.

not a perfect square. We have not further investigated the exact details behind the deviations; the observed data follows the predictions to a satisfactory degree. The relationship is nonlinear (as expected) and that means that it has to be linearized by implementing a LUT. The fact that it does not exactly follow the theoretical prediction is not crucial since it has to be calibrated with a LUT anyway.

Next we decided to stress the design by designing a TDC for some smaller range. The internal timer of the microcontroller can be clocked at 5 MHz at most (if the external crystal is 20 MHz), corresponding to a clock cycle period of 200 ns. That means that a time interval of 1 μs would only yield a few counts and a $\pm$20% uncertainty. Time stretching (or some other vernier method) will be necessary to get a "decent" resolution/uncertainty. Suppose we want to detect time intervals in the range 1–10 μs, with a precision of 0.1%. That would require a resolution of 1 ns for the "low" interval end and 10 ns for the "high" interval end. If we stretch the interval by a factor of 1000, a 10 μs pulse would be stretched to 10 ms. The maximum range of a 16-bit timer clocked at a rate of 5 MHz is 13.1 ms ($2^{16} \times 200$ ns), so that would be within our boundaries. Using design equation (10), we chose $R_1 = 90.68\ \Omega$ (measured), $C = 23.2$ nF (measured) and, using expression (11), we chose $R_2 = 470.9$ kΩ (measured). Figure 13 illustrates the calibration results of this time stretching design.

In the plot in Figure 13 we have also fitted data to the nonlinear expression (12) (using the *nlinfit()* command in MATLAB). We can rewrite expression (12) in the following way:

$$\begin{aligned} k = f(\tau) = \frac{t_d}{\tau} = \frac{\tau'}{\tau} &= -\frac{R_2C}{\tau} \times \ln\left\{\frac{V_{IL}}{U_0(1-\mathrm{e}^{-\tau/R_1C})}\right\} \\ &= \frac{R_2C}{\tau} \times \left\{\ln(1-\mathrm{e}^{-\tau/R_1C}) - \ln\frac{V_{IL}}{U_0}\right\} \\ &= \frac{a}{\tau} \times \{\ln((1-\mathrm{e}^{-\tau/b}) + c)\}, \end{aligned} \tag{13}$$

where parameters $a = R_2C$, $b = R_1C$, and $c = -\ln(V_{IL}/U_0)$. The experimental data in Figure 13 was fitted to a function described by expression (13) and this fitted curve is the continuous line in the plot in Figure 13.

V. IMPLEMETING LUT IN AN 8-BIT MICROCONTROLLER

If we multiply expression (13) by τ, we get an expression for the stretched time τ' that we measure. If we measure this time τ' using an on-chip (16-bit) timer it will produce an

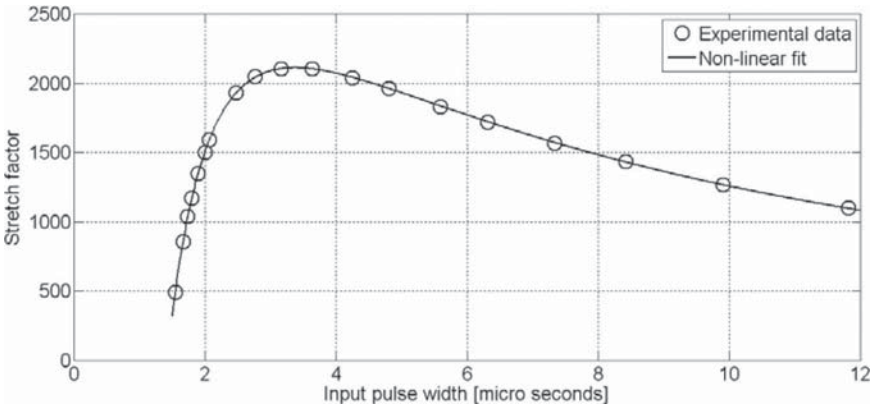

FIG. 13. Stretch factor versus pulse width.

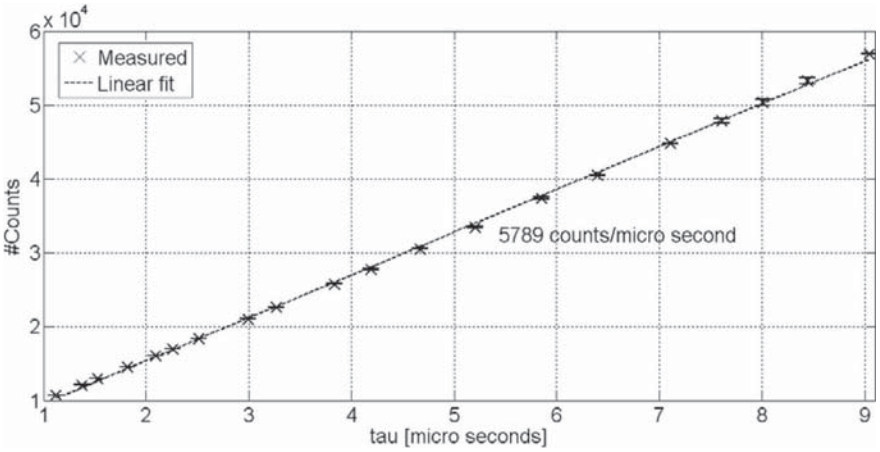

FIG. 14. TDC output count versus input time interval.

integer N. If the timer is clocked with a cycle period of t_c, then $\tau' = N \times t_c$:

$$\tau' = N \times t_c = a \times \{\ln(1 - e^{-\tau/b}) + c\}. \quad (14)$$

Parameters a, b, and c are known from the curve fitting in Figure 13. Solving for τ in Eq. (14) gives us

$$\tau = -b \times \ln(1 - e^{N \times t_c/a - c}). \quad (15)$$

We could use expression (15) to implement a LUT consisting of 2^{16} double-typed fractional τ-values. However, this would be a very memory-intensive solution and as we are implementing our solution in an embedded system, we are looking for a less memory-demanding solution. First, we want to work with integers instead of double-typed fractional (since a 16-bit integer only needs half the memory space of a double fractional). From Figure 13 we can see that the useful τ-range in the real experiment was 12 μs (measured to 12.4). Hence, if we multiply all τ-values in expression (15) by $2^{16}/12.4 = 5285$, and round to nearest integer, we convert all τ-values to into an integer range that fits into a 16-bit counter's range.

Second, we only generate the τ-values in (15) for 256 (equidistant) N-values, and our LUT will consist of these 256 τ-values only (multiplied by 5285); we use linear interpolation to find intermediate values. Notice how well this works out in an 8-bit firmware algorithm; we can use the eight most significant bits of the 16-bit timer as a pointer for the LUT and we use the eight least significant bits for the linear interpolation. In an 8-bit microcontroller they are already available in separate registers anyway. Notice that the overall linearity of the design will depend only on the accuracy of the LUT entries, the sparseness of the LUT, the resolution of the (piecewise) linear interpolation, and the curviness of the function in expression (15).[25,26]

In Figure 14 we present the output of our TDC after having implemented a linearizing LUT as described above. The looked-up counter value was transferred to a host Windows computer via an asynchronous serial interface.

The error bars in Figure 14 represent one standard deviation and varied from 10 to 400 counts due to stochastic variations in τ. (Notice that a small uncertainty $\Delta\tau$ in τ propagates into an uncertainty $k \times \Delta\tau$ in the stretched time interval.) The 5789 counts/μs indicates a potential time resolution of 0.17 ns.

VI. CONCLUSIONS

There are many situations in science and engineering that require accurate and high-resolution time measurements. This includes propagation delay measurements on integrated circuits, time domain reflectometry in cables (TDR), radar ranging, nuclear, and ballistic time of flight measurements. Also, since time can be measured more cost-efficient[24] than voltage, a lot of sensors produce pulse width modulated signals that need accurate, high-resolution time measurements.

This work has demonstrated how a sub-nano resolution TDC can be implemented in an 8-bit microcontroller with "direct" methods. We have demonstrated how pulse widths in the range 1–10 μs can be stretched by a factor of 1000–2000 times. Measuring this stretched time using a microcontroller embedded counter clocked at 200 ns yields a sub-nano resolution. The method is general and can easily be adjusted for other time intervals, however there are some limitations. First, the *minimum time interval* range that can be detected is determined by the R_1C time constant, V_{IH}, and U_0; during time τ, the capacitor C must be charged to a voltage exceeding V_{IH}. This indicates that we should chose R_1C as small as possible in order to make sure it is charged to the proper voltage level during τ. However, the *maximum time interval* is also determined by R_1C; all time intervals that saturate C (charge C to U_0) will produce the same stretched time τ' and we would not be able to uniquely resolve them.

It was observed that for repetitive measurements the produced integer varied. This is due to stochastic variations in τ and can be reduced by averaging. The overall conversion speed of the proposed TDC corresponds basically to the stretched time. In our design example the maximum stretched time was 12.4 ms and the worst-case conversion time was less than 13 ms. Notice that this conversion time is not affected by the fact that we transfer data to a host computer via an asynchronous serial link; the data transfer of the last sample takes place during the discharging time of the next sample, so no time is lost.

The proposed design in Figure 8 requires 5 digital I/O-pins. Of course, I/O-pin 4 is not really necessary if an internal timer is used to measure the stretched time. Also, the proposed design measures the *pulse width*. Some additional circuitry is needed in order to measure the time interval between two separate start and stop signals; a simple SR latch will do if the start signal is connected to the "Set" input and the stop signal is connected to the "Reset" input, the SR latch's Q output will generate a pulse that can be inserted into our TDC design in Figure 8. This is illustrated in Figure 2.

From expression (12) we can see that the stretch factor k depends on the passive components (R_1, R_2, and C) and on the U_0 and V_{IL} parameters. All the passive components are known to have temperature dependence and U_0 depends on the diode's voltage drop in forward biased mode which is also known to depend on temperature. (The temperature dependence of V_{IL} is not known.) Because of the temperature dependence of the parameters in expression (12) the stretch factor k most certainly suffers from temperature variations but that has not been examined further in this work.

[1] S. Henzler, *Time-to-Digital Converters*, Springer Series in Advanced Microelectronics Vol. 29 (Springer, New York, 2010).

[2] G. S. Jovanovic and M. K. Stojcev, "Scientific publications of the state university of Novi Pazar," Ser. A: Appl. Math. Inf. Mech **1**(1), 11–20 (2009).

[3] "Time interval measurement," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, edited by J. G. Webster (2007), see http://onlinelibrary.wiley.com/doi/10.1002/047134608X.W3989.pub2/pdf [online] (updated 13 July, 2007) [accessed December 2011].

[4] Microchip, "PIC18F2480/2580/4480/4580 Data Sheet," Microchip Inc., DS39637D, Tuscon, Arizona, 2009, see http://ww1.microchip.com/downloads/en/DeviceDoc/39637d.pdf [online] (updated: November 16, 2009) [accessed December 27, 2011].

[5] Agilent, "Fundamentals of Electronic Counters," Agilent Technologies, Application Note 200, see http://www.leapsecond.com/pdf/an200.pdf [online] (updated October 15, 2004) [accessed December 23, 2011].

[6] P. M. Levine and G. W. Roberts, "A high-resolution flash time-to-digital converter and calibration scheme," in *ITC International Test Conference 2004* (IEEE, 2004), see http://www.itcprogramdev.org/itc2004proc/papers/pdfs/0040_2.pdf [online] (updated May 16, 2006) [accessed December 19, 2011].

[7] G. W. Roberts and M. Ali-Bakhshian, IEEE Trans. Circuits Syst., II: Express Briefs **57**(3), 153–157 (2010).

[8] Agilent, "Fundamentals of Time Interval Measurements," Agilent Technologies, Application Note 200-3, see http://www.leapsecond.com/pdf/an200-3.pdf [online] (updated October 15, 2004) [accessed December 16, 2011].

[9] D. I. Porat, IEEE Trans. Nucl. Sci. **20**(5), 36–51 (1973).

[10] R. Nutt, Rev. Sci. Instrum. **39**, 1342–1345 (1968).

[11] J. Ward, Time Interval Measurement Literature Review, see http://www.rrsg.ee.uct.ac.za/members/jon/activities/timcs.pdf [online] (updated March 29, 2011) [accessed December 19, 2011].

[12] D. Cox, Implementing Ohmmeter/Temperature Sensor, Microchip Technology Inc., Application Note AN512, Chandler, Arizona, 1997.

[13] B. Merritt, MPS430 Based Digital Thermometer, Texas Instruments, Application Report SLAA038, 1999.

[14] A. Custodio, R. Bragós, and R. Pallàs-Areny, "A novel sensor-bridge-to-microcontroller interface," in *IEEE Instrumentation and Measurement Technology Conference*, Budapest, Hungary, 21–23 May 2001.

[15] J. Jordana, F. Reverter, and R. Pallàs-Areny, "Uncertainty in resistance measurements based on microcontrollers with embedded time counters," in *IMTC 2003 - Instrumentation and Measurement Technology Conference*, Vail, Colorado, USA, 20–22 May, 2003.

[16] F. Reverter and R. Pallàs-Areny, *Direct Sensor-to-Microcontroller Interface Circuits* (Maracombo, Barcelona, Spain, 2005).

[17] J. Lepkowski, Temperature Measurement Circuits for Embedded Applications, Microchip Technology Inc., Application Note AN929, Chandler, Arizona, 2004.

[18] F. Reverter and R. Pallàs-Areny, Sens. Actuators, A **127**, 74–79 (2006).

[19] L. Bengtsson, "Direct analog-to-microcontroller interfacing," Sens. Actuators, A (in press), see http://www.sciencedirect.com/science/article/pii/S0924424712001616?v=s5.

[20] R. Richey, Resistance and Capacitance Meter Using a PIC16C622, Microchip technology Inc., Application Note AN611, Chandler Arizona, 1997.

[21] F. Reverter and Ò. Casas, IEEE Trans. Instrum. Meas. **59**(10), 2763–2769 (2010).

[22] B. V. Zeghbroeck, "The p-n junction capacitance," see http://ecee.colorado.edu/~bart/book/pncap.htm [online] (updated December 1, 2000) [accessed February 1, 2012].

[23] A. W. Radun, "Non-Ideal Diode Behaviour," see http://www.engr.uky.edu/~radun/EE603/LectureNotes/DiodeChar/NonIdealDiode [online] (updated October 11, 2001) [accessed February 1, 2012].

[24] R. Pallàs-Areny and J. G. Webster, *Sensors and Signal Conditioning*, 2nd ed. (Wiley, New York, 2001).

[25] P. Hille, R. Höhler, and H. Strack, Sens. Actuators, A **44**, 95–102 (1994).

[26] S. Y. C. Catunda, O. R. Saavedra, J. V. FonsecaNeto, and R. A. Morais, "Look-up table and breakpoints determination for piecewise linear approximation functions using evolutionary computation," in *IMTC 2003, Instrumentation and Measurement Technology Conference*, Vail, Colorado, USA, 20–22 May 2003.