Implementation of Vernier TDCs in 8-bit Microcontrollers

Lars E. Bengtsson
Dept. of Physics
University of Gothenburg
SE-412 96 Gothenburg, Sweden
lars.bengtsson@physics.gu.se

***Abstract*—This paper discusses the aspects of implementing a TDC with Vernier resolution in a microcontroller system. Results will show that the proposed solution have a potential time resolution corresponding exactly to the theoretically expected resolution (equal to the time difference in the Vernier clocks' period), but also that there is an inherent sample-to-sample uncertainty due to the fact that microcontrollers cannot compare two (running) timer registers in hardware. The moment of coincidence of the Vernier clocks must be detected in software and that will generate an uncertainty in the coincidence detection that depends on the microcontroller architecture. In the design example proposed, a time resolution of 2 ns is achieved using a PIC microcontroller clocked with a 20 MHz. However, the proposed method is general and resolution is limited only to the frequency matching of the two Vernier clocks.**

Keywords—TDC; Time-to-Digital Converter; Vernier clock; microcontroller;

I. INTRODUCTION (*Heading 1*)

A Time-to-Digital Converter (TDC) is either analog or counter-based [1-3] but this work will treat only digital TDCs. A "basic" counter-based TDC simply counts the pulses from a reference clock during the start and stop interval, see Fig 1.

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 of all the power consumption increases and secondly, 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.)

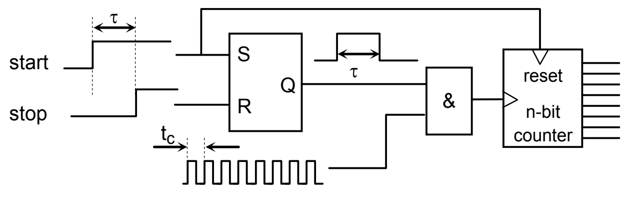

Fig. 1. Schematic principle of basic counting TDC

If we implement a counting TDC in a microcontroller-based system, the time resolution will be limited by the system's clock frequency. An 8-bit PIC-controller from Microchip with a 20 MHz crystal will internally run at 5 MHz [2]. Using its embedded 16-bit timers for time interval measurements, using the basic counting technique, would render a maximum time resolution of ±200 ns. Hence we need more advanced methods in order to achieve the time resolution characteristic of high-resolution TDCs.

There are several different ways to increase the time resolution of a digital TDC, such as time-stretching [5-10] and tapped delay lines [1,11-12] and they can in some cases be implemented in a microcontroller solution [13]. In this work, we will discuss the possibility to implement a Vernier TDC in a microcontroller design.

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, 10, 14-16]. The start and stop signals trigger one oscillator each, see Fig. 2.

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 pulses, i.e. $N_1 = N_2 = N$). From Fig. 2 it is obvious that

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

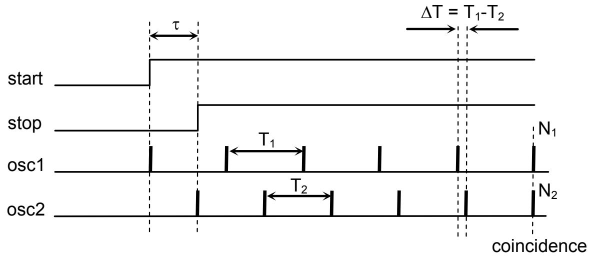

Fig. 2. The Vernier principle

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. This paper discusses how to implement such a Vernier TDC in a microcontroller, we will present some experimental results and discuss the problems involved in the implementation and their origin.

II. IMPLEMENTATION

In order to implement a Vernier TDC in a microcontroller, we need a controller with two independent timers, and at least one of them must be able to count external pulses (asynchronously); the other timer can be clocked by the internal system clock.

The theoretical time resolution corresponds to the difference in cycle periods between f_1 and f_2, i.e. $\Delta T = 1/f_1 - 1/f_2$, which indicates an uncertainty of $\pm\Delta T$ in the time interval measurement. For example, if f_1 = 4.950 MHz and f_2 = 5.000 MHz, the theoretical time resolution would be 1/4.95 – 1/5 = 2.02 ns. This should be compared to the time resolution of 200 ns that we would get if we only used the system clock and one timer.

However, in order to reach the theoretical resolution limit, we must have a perfect detection of the moment of the timers' coincidence in Fig. 2, i.e. the reading of the timers' content must not be delayed but occur exactly at the moment of coincidence. This is typically *not* possible when using microcontroller timers. The reason for that is that for a perfect detection of the moment of coincidence, the timer registers in Fig. 3 must be compared in hardware (and preferably an interrupt should be generated and the timers' content should be latched automatically).

Actually, most microcontrollers have such a mechanism; a typical general-purpose microcontroller has a "compare" mechanism that can generate an interrupt when a timer register equals the content of a compare register. This is indeed a comparison in hardware and there is no latency between the detection of the moment of coincidence and the interrupt signaling (or, the latency is predictable and deterministic). However, the comparison register must always be a "fixed" dedicated register. The Vernier TDC in Fig. 2 would require that we could compare *two running* timer registers in hardware and no microcontroller has been found that can do that. Hence, the comparison of the two timer registers must be performed in firmware as indicated in Fig. 4.

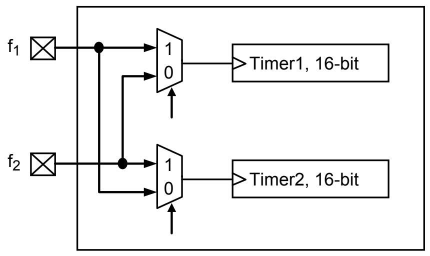

Fig. 3. A controller with two independent counter registers

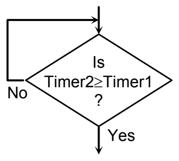

Fig. 4. Comparison in firmware of two running counter registers

For hardware-architectural reasons this kind of algorithms will necessarily consume more than one instruction cycle (ic) and if one of the clocks (f_2) runs on the system clock (Timer2 is updated synchronously on every ic) the moment of timers coincidence may statistically occur at any time during the execution of the algorithm; if the algorithm consumes m ics, the probability of detecting the exact moment of coincidence is only $1/m$, see Fig. 5.

To (partly) overcome this problem, the algorithm must not try to detect the moment of coincidence (Timer2 = Timer1), but must rather detect the moment when the content of Timer2 equals *or exceeds* Timer1 (Timer2 ≥ Timer1). Consequently, if we read Timer2 (= N_2) when the condition in Fig. 4 is true, the actual moment of coincidence may have occurred at any moment in the interval [N_2–m..N_2] (where m is the number of ics consumed by the firmware algorithm in Fig. 4). Since this will have a uniform distribution, the best estimator is the mid-range value:

$$\hat{N}_{coinc} = N_2 - \frac{m}{2} \qquad (2)$$

III. METHODS AND MATERIAL

The Vernier TDC idea described above was implemented in an 8-bit PIC18F controller from Microchip (PIC18F4580). It was clocked by a 20 MHz crystal and since the internal system clock runs at ¼ of the external crystal frequency, the internal system speed is 5 MHz. This system clock was used to increment one timer. The other timer was incremented asynchronously by an external 4.950 MHz oscillator, see Fig. 6.

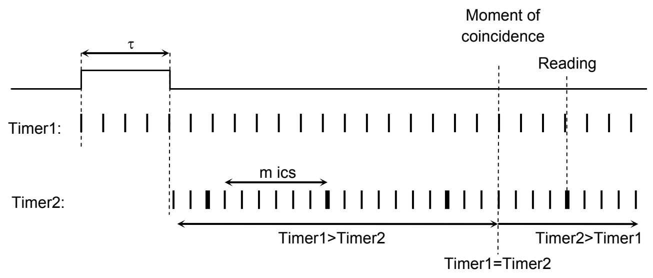

Fig. 5. Registers are compared only on every mth ic

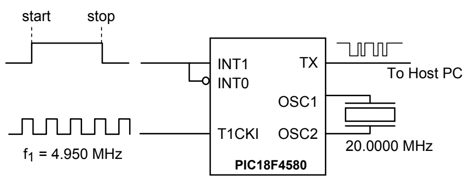

Fig. 6. Schematic drawing of implementation

The edges of the pulse to be measured were used to trigger external interrupts; the positive edge triggered a context switch that started Timer1 and the negative edge triggered a context switch that started Timer2. The timer registers were compared in firmware and data were sent to a host Windows PC via an asynchronous serial data link.

From equation (2) it is clear that the uncertainty of the moment of coincidence depends on *m*, and for that reason it is important to design the firmware in such a way that the time consumed by the algorithm in Fig. 4 is as short as possible (minimize *m*). For that reason the algorithm in Fig. 4 was written in assembler while the rest of the firmware was written in C (using a C-compiler from Hi-TECH). Even so, the minimum number of clock cycles that could execute the algorithm in Fig. 4 was 30 ics. Also, by waiting for the start and stop signals in idle mode, the microcontroller's interrupt latency is minimized [17].

Since we need to reset two 16-bit timers in an 8-bit environment, it will take two ics to do that.

First we clear a buffer register which will be automatically loaded into the timer's eight most significant bits (msb) when we clear the eight least significant bits (lsb) on the next ic [4].

However, we can clear the buffer registers before we put the CPU to sleep and if we clear the eight lsbs as the first instruction after wake-up, the 16-bit timer will be cleared immediately after wake-up. Notice that this is executed even before the ISR (Interrupt Service Routine) is called; the controller *first* executes *one* instruction after wake-up (the one right after the *sleep*-instruction) and *then* does the context switch.

Still the ISR needs to be served and even if we only clear an interrupt flag in the ISR, there is always some ISR overhead that will consume some ics. This is compiler dependent and the Hi-TECH C-compiler used in this work (PICC-18 Lite) needs 64 ics to execute the ISR and return. Hence, there is no delay in the clearing of the timer registers, but due the ISR overhead in the compiler used, there is a limit to the minimum time interval that can be detected. If the system clock runs at 5 MHz, this minimum limit is 12.8 μs (64/5 μs).

IV. Experimental Results

The TDC in Fig. 6 was tested and calibrated by using start-stop pulses of well-known durations. A HP8013B pulse generator was used for this purpose. The calibration result is illustrated in Fig. 7. The calibration samples in Fig. 7 represent the average of 416 samples.

In order to get an idea of the precision of the sample-to-sample values, histograms were plotted for a large number of samples corresponding to the same input pulse duration. Fig. 8 illustrates the histogram of 416 samples for a pulse duration of 52.58 μs.

V. Discussion

The gradient of 486 counts/μs in Fig. 7 corresponds exactly to the theoretically predicted resolution of 2.02 ns (per count). The upper range limit was measured to 131 μs and this also agrees very well with the theoretical value ($2^{16} \times 2.02$ ns = 132 μs). The lower limit depends on the ISR overhead as described in the previous section and was measured to 16 μs which is slightly larger than the theoretical value (12.8 μs). The sample distribution in Fig. 8 has a standard deviation of 41 counts. This sample-to-sample uncertainty has two main contributions:

1. The standard uncertainty of the pulse width was 0.02 μs (measured with a digital Tektronix oscilloscope, TDS5000). With a sensitivity of 486 counts, this uncertainty corresponds to 10 counts.
2. Look at Fig. 6. The timer incremented by the system clock is controlled by the external 20 MHz crystal which is stable on a ppm-level. However, the other timer is incremented by the external 4.95 MHz clock source. This clock source was generated by an Agilent 33220A function generator and its frequency stability was measured (also with a Tektronix TDS5000 oscilloscope). The frequency was measured to 4.950 ± 0.007 MHz (1-sigma interval). Hence, there is a non-negligible frequency uncertainty in the update rate of Timer1 of 0.14%. The mean value of the data in Fig. 8 is 25433 counts and a frequency uncertainty of 0.14% corresponds to 36 counts.

If we assume these contributions to be independent, the total expected uncertainty is [18]:

$$\sigma = \sqrt{\sigma_1^2 + \sigma_2^2} = \sqrt{10^2 + 36^2} = 37 \qquad (3)$$

which agrees very well with the experimentally observed value of 41.

The shape of the distribution histogram also needs some comments. It is interesting to see that some values seem to be missing. This is expected; since executing *if (Timer2>= Timer1)* in firmware consumes 30 ics, only every 30th count value should really appear at all, but due to the noise in both the pulse width and the external clock source, some intermediate values will appear. Also, we really don't know the distribution of the pulse width noise and the external clock frequency.

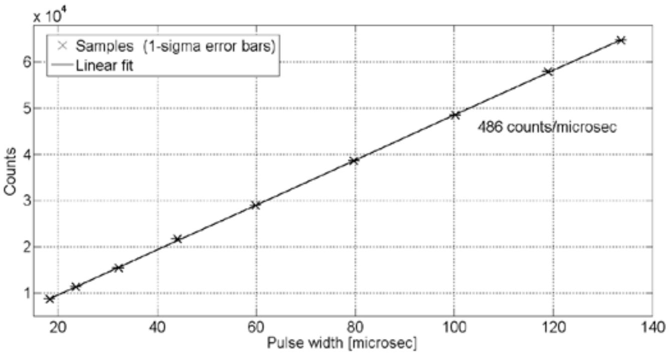

Fig. 7. Calibration diagram for the microcontroller TDC

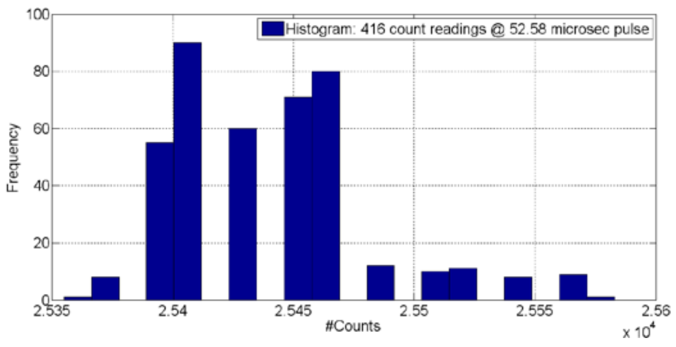

Fig. 8. Typical data distribution diagram

VI. CONCLUSIONS

This work has demonstrated how to implement a high-resolution TDC in a simple 8-bit microcontroller. The controller needs to have two counters/timers and a low-power idle mode that allows peripherals to run during hibernation. The design proposed here improves the TDC resolution by a factor of 100 (from 200 ns to 2 ns) but the method is general and is limited only by the frequency difference of the Vernier clocks. However, due to uncertainties caused by firmware overhead and Vernier clock jitter, the 1-shot precision is only ±30 ns. Very accurate Vernier clocks are necessary in order to improve the precision which suggests that both clocks must be crystal based. This may be a problem though, because if both clocks are crystal based, only certain frequencies are available and the time resolution will depend on what quartz crystals are available. This might lead to a trade-off between resolution and precision but remember that due to the firmware induced uncertainty, averaging will most likely be necessary anyway. Hence, if we use a tunable external clock for Timer2, the trade-off is really between resolution and bandwidth.

In the proposed design, the microcontroller enters the low-power sleep mode and just waits for the start and stop edges. If the design requires the controller to be in an active mode, the minimum pulse width limit will be affected, since the registers in this case must be cleared in the ISR and that adds at least 64 ics to the minimum pulse width that can be detected.

The main advantage of the proposed TDC implementation is the performance/price ratio. The design has not been tested for temperature dependence, but can most likely be designed for excellent temperature (in)dependence by using crystal based oscillators.

There are some things in this design that could certainly be improved in the future. First of all, a stable clock source for Timer1 is necessary and a high-resolution precise counter should be used to determine the exact pulse widths instead of a digital oscilloscope. More precise measurements of the deviation from linearity in Fig. 7 need also be performed.

REFERENCES

[1] Henzler, S. "Time-to-Digital Converters", Springer Series in Advanced Microelectronics 29, doi 10.1007/978-90-481-8628-0__2, 2010.

[2] Jovanovic, G.S. and Stojcev, M.K., Appl. Math. Inform. and Mech. **vol 1**, 1(2009), pp 11-20.

[3] Webster, J.G. "Time Interval Measurement." Wiley Encyclopedia of Electrical and Electronics Engineering. [online] (Updated 13 July, 2007). Available at <http://onlinelibrary.wiley.com/doi/10.1002/047134608X.W3989.pub2/pdf> [Accessed December 2011].

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

[5] Agilent Technologies. "Fundamentals of Electronic Counters", Application Note 200, [online] (Updated October 15, 2004). Available at <http://www.leapsecond.com/pdf/ an200.pdf> [Accessed December 23, 2011].

[6] Nutt R., Rev. Sci. Instrum. **39**, pp. 1342-1345, 1968.

[7] Ward, J. Time Interval Measurement Literature Review. [online] (Updated March 29, 2011). Available at <http://www.rrsg.ee.uct.ac.za/members/jon/activities/timcs.pdf> [Accessed December 19, 2011].

[8] Kalisz, J., Pawlowski, M. and Pelka, R., J. Phys. E: Sci. Instrum. **vol 18**, pp. 444-452, 1985.

[9] Räisänen-Ruotsalainen, E., Rahkonen, T. and Kostamovaara, J., IEEE Journal of Solid-State Circuits, **vol. 35**, no. 10, pp. 1507-1510, October 2000.

[10] Agilent Technologies. "Fundamentals of Time Interval Measurements", Application Note 200-3, [online] (Updated October 15, 2004). Available at <http://www.leapsecond.com/pdf/an200-3.pdf> [Accessed December 16, 2011].

[11] Levine, P.M. and Roberts, G.W. "A high-resolution flash time-to-digital converter and calibration scheme." In: ITC International Test Conference 2004. [online] (Updated May 16, 2006). Available at <http://www.itcprogramdev.org/ itc2004proc/papers/pdfs/0040_2.pdf> [Accessed December19, 2011].

[12] Roberts, G.W. and Ali-Bakhshian, M., IEEE Transactions on Circuits and Systems-II: Express Briefs, **vol. 57**, no. 3, March 2010.

[13] Bengtsson, L. Rev. Sci. Instr., **83**, 045107 (2012), doi 10.1063/1.3700192.

[14] Porat D.I., IEEE Transactions on Nuclear Science, **vol. 20**(5), pp. 36-51, October 1973.

[15] Kang, X., Liu, Y, Sun, X, Wang, S., Yan, X., Zhang, Z., Wu, Z. and Jin, Y., 2008 International Conference on BioMedical Engineering and Informatics, doi: 10.1109/BMEI.2008.350, 2008.

[16] Amiri, A. M., Boukadoum, M and Khouas, A., IEEE Transactions on Instrumentation and Measurement, **vol. 58**, no 3, March 2009.

[17] Reverter F. and Pallàs-Areny R., 2006. *Elsevier: Sensors and Actuators A*, **127**, pp 74-79, 2006.

[18] Adams, T. M., 2002. "G104-A2LA Guide for Estimation of Measurement Uncertainty in Testing", [online] (Updated February 29, 2008) Available at <http://www.a2la.org/guidance/ est_mu_testing.pdf> [Accessed January 2, 2012].