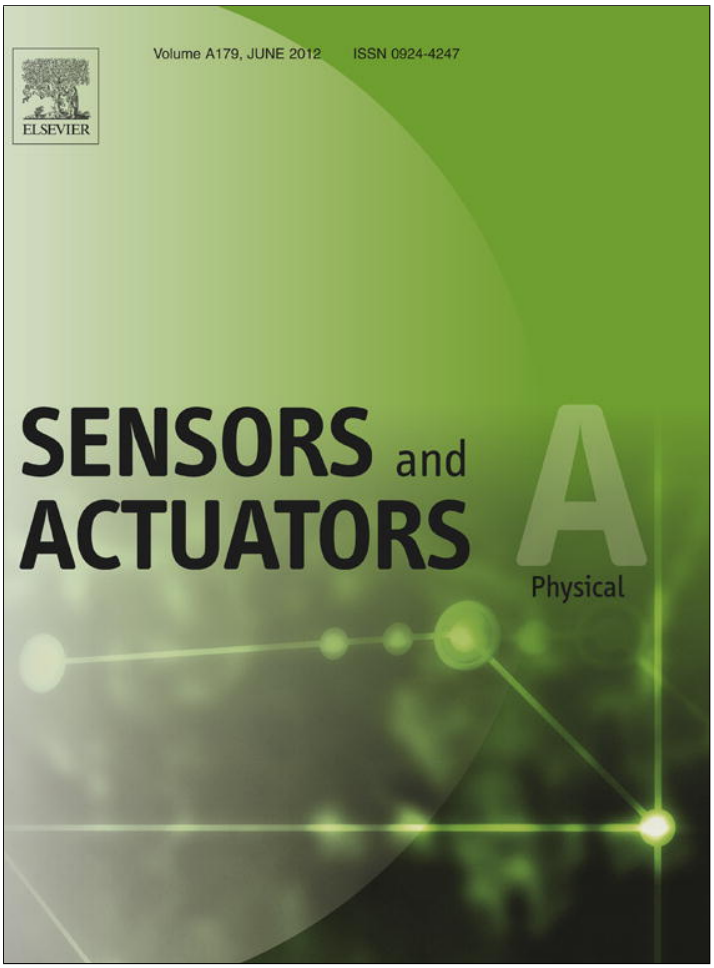

Sensors and Actuators A 179 (2012) 105–113

Contents lists available at SciVerse ScienceDirect

Sensors and Actuators A: Physical

journal homepage: www.elsevier.com/locate/sna

Direct analog-to-microcontroller interfacing

Lars Bengtsson

University of Gothenburg, Physics Department, SE-412 96 Gothenburg, Sweden

ARTICLE INFO

Article history:
Received 14 November 2011
Received in revised form 20 February 2012
Accepted 28 February 2012
Available online 7 March 2012

Keywords:
Analog
Sensor interfacing
Microcontroller
Embedded system

ABSTRACT

This paper will demonstrate how signals from analog sensors can be directly interfaced to any digital embedded system even though they may not be equipped with an on-chip ADC (Analog-to-Digital Converter), comparator or OP amp (Operational Amplifier). With only two resistors and one capacitor, we will present a solution that allows analog voltages to be measured directly using only a few digital I/O-pins. The digital target system requirements are minimized and limited to only two digital I/O-pins (with tri-state capability). No ADC, comparators, timers or capture modules are necessary. The extremely modest hardware requirements make it a suitable solution also for CPLDs/FPGAs (Complex Programmable Logic devices/Field Programmable Gate Arrays). Since this solution will allow even the simplest embedded system to be interfaced to analog voltage sensors, it has the potential of reducing design costs considerably. The proposed design is for DC or low-frequency signals and compared to other similar solution, this design needs no embedded analog blocks at all.

1. Introduction

The techniques of interfacing analog sensors directly to digital embedded systems without the use of embedded ADCs or complex signal conditioning electronics was developed in the mid-1990s. These techniques may be divided into two categories. One class of interfaces focused on passive sensors (i.e. resistive, capacitive and bridge sensors). Pioneering work was presented by Cox (1997) [1], Richey (1997) [2], Baker (1999) [3] and Bierl (1996) [4].

The other class is focused on sensors with analog voltage outputs and Peter et al. [5] showed in 1998 that these sensors could be interfaced without using the embedded SAR ADC (Successive Approximation Register) of a typical microcontroller. If only the controller has an embedded comparator, a ΣΔ ADC (Sigma Delta) can be implemented with only a few passive components. This has later been confirmed and demonstrated by several, both in microcontrollers [6–8] and in FPGAs (with embedded or external comparators) [9,10].

Fig. 1 illustrates the basic idea of the first class of "non-ADC" interfaces for passive resistive sensors [1].

The analog-to-digital conversion is a multi-step process: in step one, the capacitor is charged via I/O-pin 3. I/O-pin 3 is configured as output and set high, while the other pins are set to inputs (High-Z). In the next step, I/O-pin 3 is configured as input (High-Z) while I/O-pin 2 is configured as output and set low; the capacitor will discharge through R_S (=the sensor) until it reaches V_{IL} (input logic low threshold) of I/O-pin 3 and a firmware variable is incremented during the discharging in order to measure the discharging time. This produces an integer N_S, proportional to the sensor's resistance. The procedure is repeated for the calibration resistor R_C, producing an integer N_C, proportional to R_C. From this data, the sensor resistance can be estimated independently of the capacitor value C [1]:

$$\hat{R}_S = \frac{T_S}{T_C} R_C = \frac{N_S}{N_C} R_C \quad (1)$$

Corresponding techniques have been developed for capacitive sensors [2,11–14], differential capacitive sensors [15] and resistive bridge sensors [16–18].

The implementation of ΣΔ ADCs in digital embedded systems are based on the circuit in Fig. 2.

In Fig. 2, the capacitor C represents the integrator in a traditional ΣΔ ADC [5,19]. When the digital I/O pin is set high, the capacitor is charged and when reset, the capacitor is discharged. This is controlled in firmware by polling the comparator's output. If the comparator's output is high the I/O-pin is reset and if it is low the I/O-pin is set [21]. If the input voltage $x(t)$ is high, more 0s are required to discharge the capacitor and if the input voltage is low fewer 0 s are required to discharge it. The result is a control loop where the comparator's positive input is regulated to $V_{DD}/2$ by varying the density of 1 s and 0 s on the digital I/O-pin. Since the comparator's output is inverted compared to the digital I/O-pin, the density of 1 s in the bitstream from the comparator's output is proportional to the input analog voltage $x(t)$. This bitstream is then decimated in firmware in order to output an n-bit integer.

Hence the embedded system circuit in Fig. 2 is capable of handling analog voltages without having an embedded ADC and it has

E-mail address: lars.bengtsson@physics.gu.se

0924-4247/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.sna.2012.02.048

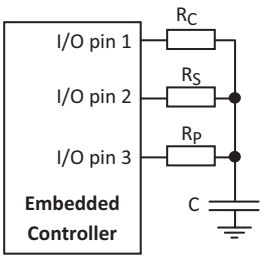

Fig. 1. The one-point calibration technique.

all of the advantages of any ΣΔ ADC; it reduces the quantization noise by *oversampling* and *noise shaping* [19,20]. However, it has disadvantages too. It is slow (limited bandwidth) and most of all, it only works for embedded systems with integrated comparators, which indicates that it requires a relatively advanced (expensive) microcontroller and excludes inherently digital systems such as CPLDs and FPGAs.

The solution presented in this work, uses only two I/O-pins like the ΣΔ ADC in Fig. 2, but has the advantage of not requiring a comparator; the technique can be implemented also in FPGAs.

The rest of this paper is organized as follows: Section 2 describes the hardware and firmware solutions, Section 3 analyses the hardware and firmware in detail and we derive design rule expressions as well as a complete theoretical description of the system. At the end of Section 3 we will demonstrate how well experimental data agrees with theoretical predictions. Section 4 describes the equipment and measurement methods that were used to perform this work and a detailed schematic of the hardware. Section 5 analyses the design from an uncertainty point of view and we demonstrate how the uncertainty of all the system parameters propagate into an uncertainty in the analog voltage that is measured. Finally, Section 6 draws some important conclusions about the design.

2. Direct analog-to-digital interface

2.1. Hardware

Fig. 3 illustrates the suggested solution.

In Fig. 3, A_{in} represents the analog voltage generated by the sensor. This circuit also requires that the embedded system's I/O-ports have tri-state capability. The capacitor is first charged to A_{in} by configuring the I/O-pins as High-Z inputs. Depending on the voltage level of A_{in}, I/O-pin 2 will reach the input logic high threshold (V_{IH}) or not. If C is charged to a level higher than V_{IH}, then the capacitor is discharged through R_1 to V_{IL} (=the input logic low threshold) by configuring I/O-pin 1 as digital out, logic low and the discharging time is proportional to A_{in}. If the charging of C does not reach the V_{IH} level on I/O-pin 2, we instead measure the time it takes to charge

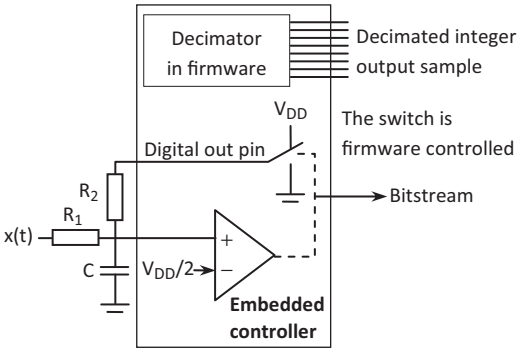

Fig. 2. Implementing a ΣΔ ADC in an embedded controller.

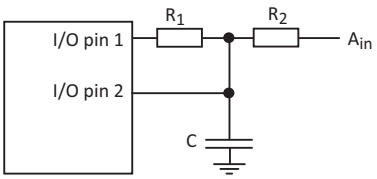

Fig. 3. Analog-to-digital converter.

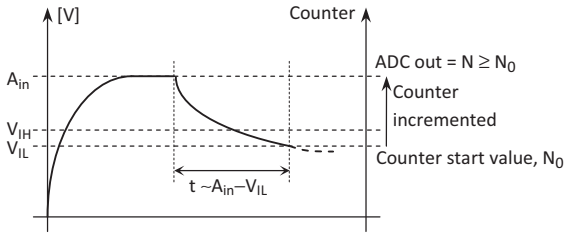

Fig. 4. Case 1: $A_{in} \geq V_{IH}$.

it all the way up to the V_{IH} level (i.e. from A_{in} to V_{IH}) by configuring I/O-pin 1 as digital out, logic high. In the latter case, the charge time is proportional to $V_{IH} - A_{in}$.

This provides all the hardware necessary for an n-bit ADC with a range of 0–V_{DD} and a (theoretically) arbitrary resolution; the resolution is determined by the number of bits of the counting variable in firmware, the counter's speed and the R_1C time constant in Fig. 3.

2.2. Firmware

When the capacitor has been charged to A_{in}, we have one of two possible situations; either the voltage potential on I/O-pin 2 is greater than (or equal to) V_{IH} (input high level threshold) or it is not. In software we have a counting variable which is initiated to a start value N_0 and if $A_{in} \geq V_{IH}$, this variable is *incremented* until C is *discharged* to V_{IL}. If, on the other hand, $A_{in} < V_{IH}$, the counting variable is *decremented* from N_0 until C is *charged* to V_{IH}. The charging/discharging is implemented by reconfiguring I/O-pin 1 to output and setting it high or low.

These two possible situations are illustrated in Figs. 4 and 5.

Fig. 6 illustrates the firmware in flowchart form. After a short initialization, there is an infinite loop that performs the measurement. The counter variable is initiated and then I/O-pins 1 and 2 are configured to High-Z so that the capacitor can be charged to A_{in}. The charging time is constant (=$5R_2C$). After the charging time has expired, the firmware decides whether or not the capacitor was charged to V_{IH} or not; this will decide whether I/O-pin 1 is set high or low when configured as an output pin. If set high, the counter variable is decremented until I/O-pin 2 reaches V_{IH}, if set low, the counter variable is incremented until I/O-pin 2 reaches the V_{IL} level. Finally, the data is transferred to a host Windows computer via an asynchronous serial link in the "ADC out" box. We will comment this in more detail in Section 3.

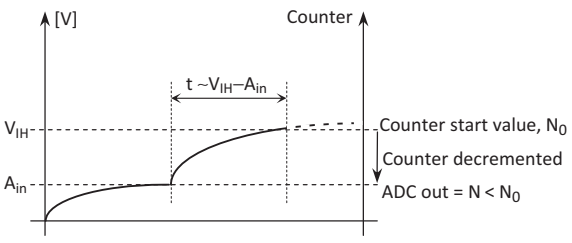

Fig. 5. Case 2: $A_{in} < V_{IH}$.

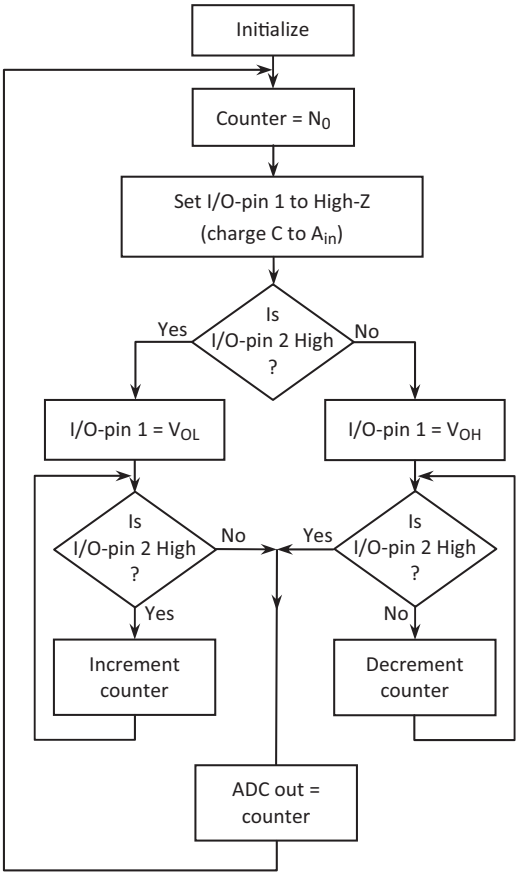

Fig. 6. Firmware flowchart.

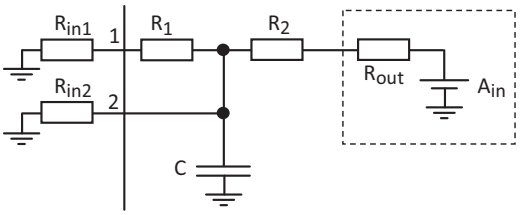

Fig. 7. System model during first stage.

3. System analysis

3.1. Hardware

During the first stage, when the capacitor is charged, both I/O-pins are configured as inputs represented by resistors R_{in1} and R_{in2} to ground. Also, according to Thevenin's theorem we can represent the signal source as an ideal signal source and a series resistance R_{out}. Fig. 7 illustrates the system model during this stage.

If we first assumes $R_{in1} = R_{in2} = \infty$ and $R_{out} = 0$, the model reduces to the circuit in Fig. 8.

$U_c(t)$ increases exponentially and is charged to 99.3% of A_{in} in a time equal to $5R_2C$. In our design, $R_{out} = 50\,\Omega$ (nominal) and $R_2 = 10.023\,k\Omega$ (measured), so excluding R_{out} from our model introduces a 0.5% error in the charging time constant. Also, R_{in1} and R_{in2} were measured to 1.012 MΩ (as described in [23]). This will cause a small leakage current through the I/O-pins of the order of a few micro amps (depending on A_{in}). However, even if we include non-ideal values of R_{out}, R_{in1} and R_{in2}, the capacitor will still be charged to >99% of A_{in} in a time $5R_2C$.

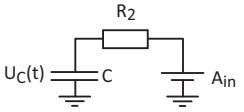

Fig. 8. Reduced system model.

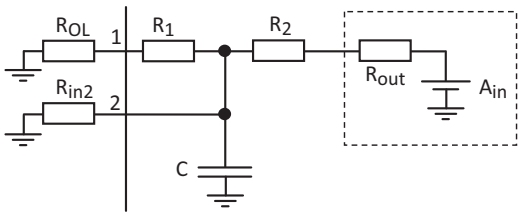

Fig. 9. System model during discharging.

Depending on whether C is charged to V_{IH} or not, the capacitor will be either discharged or charged during phase 2. Assume first that C was charged to a voltage $\geq V_{IH}$. I/O-pin 1 will be reconfigured to an output port and set low which will discharge the capacitor. Fig. 9 illustrates the system model during the discharging.

R_{OL} is the I/O-pin's output resistance when set low. If we first assume R_{OL} and $R_{out} = 0$ and $R_{in2} = \infty$, the system model is reduced to Fig. 10.

The capacitor will be discharged until $U_C(t)$ reaches the input logic low threshold V_{IL} of I/O-pin 2. From Fig. 10 we can see that the sinking current i_1 equals $i_2 + i_3$:

$$i_2 + i_3 = i_1 \tag{2}$$

$$\frac{A_{in} - U_c(t)}{R_2} - C\frac{d}{dt}U_c(t) = \frac{U_c(t)}{R_1} \tag{3}$$

$$\frac{d}{dt}U_c(t) + \frac{1}{C}\left(\frac{1}{R_1} + \frac{1}{R_2}\right)\cdot U_c(t) = \frac{A_{in}}{CR_2} \tag{4}$$

This is a first order differential equation with the following solution:

$$U_c(t) = \frac{R_1}{R_1 + R_2}A_{in} + U_0 \cdot e^{-(1/c)((1/R_1)+(1/R_2))\cdot t} \tag{5}$$

Since $U_C(0) = A_{in}$, we can easily find U_0:

$$A_{in} = \frac{R_1}{R_1 + R_2}A_{in} + U_0 \Rightarrow U_0 = \frac{R_2}{R_1 + R_2}A_{in} \tag{6}$$

$$U_c(t) = \frac{A_{in}}{R_1 + R_2}(R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t}) \tag{7}$$

$U_C(t)$ will reach the input logic low threshold of I/O-pin 2 ($=V_{IL}$) after time t_d:

$$t_d = -\frac{1}{(1/C)((1/R_1) + (1/R_2))} \cdot \ln\left\{\frac{1}{R_2}\left(\frac{V_{IL}(R_1 + R_2)}{A_{in}} - R_1\right)\right\} \tag{8}$$

We need expression (8) to predict the range of expected discharging times in order to be able to design the firmware properly. We get the maximum t_d by inserting $A_{in} = V_{DD}$. The firmware counter will produce a value N_d proportional to t_d. We will analyze

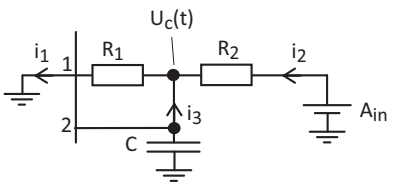

Fig. 10. A simplified system model at discharging.

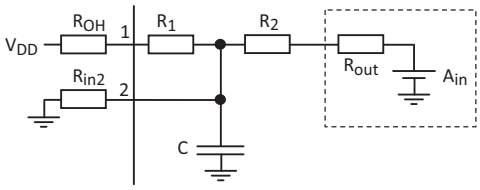

Fig. 11. System model during phase 2 when $A_{in} < V_{IH}$.

the relationship between t_d and N_d later in the firmware section. Having measured t_d, we can find A_{in}:

$$A_{in} = V_{IL} \cdot (R_1 + R_2) \cdot (R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t_d})^{-1} \tag{9}$$

It is important that resistors R_1 and R_2 are dimensioned so that I/O-pin 2 can reach a potential less than V_{IL}. That means that $U_C(t=\infty)$ in (7) should be less than V_{IL}, and this must hold for any A_{in} in the range V_{IH}–V_{DD}. The worst case is $A_{in} = V_{DD}$:

$$\frac{V_{DD}}{R_1 + R_2}(R_1 + 0) \le V_{IL} \Rightarrow \frac{R_1}{R_1 + R_2} \le \frac{V_{IL}}{V_{DD}} \tag{10}$$

(10) is our first design rule that we need to keep in mind when designing the hardware. If we compare Figs. 9 and 10, we can see some consequences of our idealized model. We already know that R_{out} in our example is about 0.5% of R_2 and R_2 should really be replaced with $R_2 + R_{out}$ in expressions (6)–(10). Neglecting R_{out} means we have a 0.5% error in R_2 in expressions (6)–(10). If we include R_{OL}, the discharging time will *increase*. However, if we also include R_{in2}, there will be a leakage current through I/O-pin 2, which will *decrease* the discharging time. The effects of R_{OL} and R_{in2} will partly cancel. If we include R_{OL} and R_{in2}, the discharging resistor is not just R_1; it will be $(R_1 + R_{OL})$ in parallel with R_{in2}. R_{OL} was measured to 21.79 Ω. In our design we used $R_1 = 2191\ \Omega$ (measured) and including $R_{OL} = 21.79\ \Omega$ and $R_{in2} = 1.012\ \text{M}\Omega$, changes the discharging resistance value from $R_1 = 2191\ \Omega$ to $(R_1 + R_{OL})/R_{in2} = 2208\ \Omega$. This is an error of 0.8% and to get an idea of how this transfers into an error in the discharging time, we treat this error as an uncertainty in R_1, i.e. $R_1 = 2191 + dR$, where $dR = 2208 - 2191 = 17\ \Omega$. We get the corresponding uncertainty in the discharging time by differentiating expression (8):

$$t_d = f(R_1) \Rightarrow dt_d = f'(R_1)dR_1 \tag{11}$$

We differentiated expression (8) by using the web-based symbolic computational engine from WolframAlpha (www.wolframalpha.com) and by inserting measured parameter values from our design example, we found that an 0.8% error in the discharging resistance transfers into a 1.3% error in the maximum discharging time $t_{d,\max}$.

If the capacitor C is not charged to a voltage $\ge V_{IH}$, during phase one, I/O-pin 1 will be configured as a digital output pin and set high, see Fig. 11.

Again, we simplify the model by assuming that R_{OH} and $R_{out} = 0$ and $R_{in2} = \infty$. That gives us the model in Fig. 12.

The capacitor will now be charged until $U_C(t)$ reaches the input logic high threshold V_{IH} of I/O-pin 2. From Fig. 12 we can see that the sourcing current i_1 equals $i_2 + i_3$:

$$i_2 + i_3 = i_1 \tag{12}$$

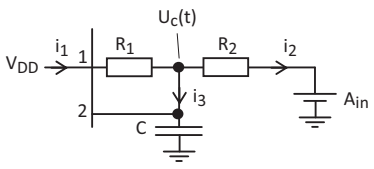

Fig. 12. A simplified system model when charging.

$$\frac{U_c(t) - A_{in}}{R_2} - C \cdot \frac{d}{dt}U_c(t) = \frac{V_{DD} - U_c(t)}{R_1} \tag{13}$$

$$\frac{d}{dt}U_c(t) - \frac{1}{C}\left(\frac{1}{R_1} + \frac{1}{R_2}\right) \cdot U_c(t) = \frac{1}{C}\left(\frac{V_{DD}}{R_1} + \frac{A_{in}}{R_2}\right) \tag{14}$$

This is a first order differential equation with the following solution:

$$U_c(t) = \frac{(V_{DD}/R_1) + (A_{in}/R_2)}{(1/R_1) + (1/R_2)} + U_0 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t} = \frac{V_{DD}R_2 + A_{in}R_1}{R_1 + R_2} + U_0 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t} \tag{15}$$

Since $U_C(0) = A_{in}$, we can easily find U_0:

$$A_{in} = \frac{V_{DD}R_2 + A_{in}R_1}{R_1 + R_2} + U_0 \Rightarrow U_0 = \frac{R_2}{R_1 + R_2}(A_{in} - V_{DD}) \tag{16}$$

$$A_{in} \Rightarrow U_c(t) = \frac{R_2}{R_1 + R_2}V_{DD} + \frac{R_1}{R_1 + R_2}A_{in} + \frac{R_2}{R_1 + R_2}(A_{in} - V_{DD}) \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t} = \tag{17}$$

$$A_{in} = \frac{R_2}{R_1 + R_2}V_{DD}(1 - e^{-(1/C)((1/R_1)+(1/R_2))\cdot t}) + \frac{A_{in}}{R_1 + R_2}(R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t}) \tag{18}$$

$U_C(t)$ will reach the input logic high threshold of I/O-pin 2 ($=V_{IH}$) after some time t_c:

$$t_c = -\frac{1}{(1/C)((1/R_1) + (1/R_2))} \cdot \ln\left\{\frac{V_{IH} - (R_2/(R_1 + R_2))V_{DD} - (R_1/(R_1 + R_2))A_{in}}{(R_2/(R_1 + R_2))(A_{in} - V_{DD})}\right\} = \tag{19}$$

$$t_c = -\frac{1}{(1/C)((1/R_1) + (1/R_2))} \cdot \ln\left\{\frac{V_{IH}(R_1 + R_2) - R_2V_{DD} - R_1A_{in}}{R_2(A_{in} - V_{DD})}\right\} \tag{20}$$

We get the maximum t_c by setting $A_{in} = 0$. Having measured t_c, we can find A_{in} by setting $U_C(t) = V_{IH}$ and solving for A_{in} in (18):

$$A_{in} = \left(V_{IH} - \frac{R_2}{R_1 + R_2}V_{DD}(1 - e^{-(1/C)((1/R_1)+(1/R_2))\cdot t_c})\right) \cdot (R_1 + R_2) \cdot (R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot t_c})^{-1} \tag{21}$$

It is important that resistors R_1 and R_2 are dimensioned so that the final value of $U_C(t)$ is $\ge V_{IH}$. The worst case is when $A_{in} = 0$. Inserting $A_{in} = 0$, $t = \infty$ and $U_C(t) \ge V_{IH}$ into (18) gives us

$$\frac{R_2}{R_1 + R_2}V_{DD} \ge V_{IH} \Rightarrow \frac{R_2}{R_1 + R_2} \ge \frac{V_{IH}}{V_{DD}} \tag{22}$$

Compare this expression with expression (10). (22) is our second design rule.

If we compare Figs. 11 and 12 we can get an idea of the consequences of our simplified model. R_{OH} was measured to 57.15 Ω. As opposed to the discharging case, including R_{OH} and R_{in2} will not cancel each other. Including R_{OH} *increases* the charging time and if we assume that $R_{in2} < \infty$, then some current is leaked through I/O-pin 2 and that also *increases* the charging time. However, if we *also* include R_{out}, the current i_2 (through R_2) decreases and that means that i_3 increases which will *decrease* the charging time (since more current is directed to capacitor C). R_{out} will compensate (partly) for R_{OH} and R_{in2}. In order to quantify this contribution, we could treat it

principally in the same way as we treated the same problem in the discharging case, i.e. treating R_{OH} as an uncertainty in R_1, differentiating expression (20) with respect to R_1 and then use (11) to find dt_c. We have not performed these calculations for this case for the following reasons: (1) the calculations follow the same principle as previously demonstrated (but are somewhat more complicated), (2) R_{OH} and R_{out} are almost equal (57.15 and 50 Ω, respectively). That means that the decrease in i_1 due to R_{OH} is to the most part compensated by a decrease in i_2 due to R_{out} and the total influence on i_3 will be very small. (3) The leakage current through R_{in} is proportional to $U_C(t)$ which in this case is $\leq A_{in}$. In this case we are at the "low end" of A_{in}, $[0 \ldots V_{IH}]$, and therefore the leakage current is less than 1.2 μA (and in the sub-micro amp range for most cases). In total, our simplified model is more accurate in the charging case than in the discharging case.

Let's also take a closer look at design Eqs. (10) and (22). Both actually set at condition for the ratio R_2/R_1. (10) can be rewritten as

$$\frac{R_2}{R_1} \geq \frac{V_{DD}}{V_{IL}} - 1 \tag{23}$$

and we can write (22) as

$$\frac{R_2}{R_1} \geq \frac{V_{IH}}{V_{DD} - V_{IH}} - 1 \tag{24}$$

Only one of these conditions needs to be considered and is determined by the sum of V_{IL} and V_{IH}: If $V_{IL} + V_{IH} < V_{DD}$, then only design rule (10) = (23) needs to be considered, since

$$\underbrace{\frac{V_{DD}}{V_{IL}} - 1}_{\uparrow (23)} = \frac{V_{DD} - V_{IL}}{V_{IL}} > \frac{V_{IH}}{V_{IL}} > \underbrace{\frac{V_{IH}}{V_{DD} - V_{IH}}}_{\uparrow (24)} \tag{25}$$

On the other hand, if $V_{IL} + V_{IH} > V_{DD}$, then only design rule (22) = (24) needs to be considered, since

$$\underbrace{\frac{V_{IH}}{V_{DD} - V_{IH}}}_{\uparrow (24)} > \frac{V_{DD} - V_{IL}}{V_{DD} - V_{IH}} > \frac{V_{DD} - V_{IL}}{V_{IL}} = \underbrace{\frac{V_{DD}}{V_{IL}} - 1}_{\uparrow (23)} \tag{26}$$

Hence, the $V_{IL} + V_{IH}$ sum should be determined first because this determines which one of the design rules (10) or (22) we need to consider.

3.2. Firmware

The firmware was written in C-code, according to the flowchart in Fig. 6. The critical parts are the incrementing and decrementing of the counter variable. If we aim for 12 bits resolution, the counter variable range is 0–4095. We should divide this interval properly between the incrementing and decrementing situations in Figs. 4 and 5. In other words, we need to find N_0. To find N_0, we first find the maximum values of t_d and t_c in expressions (8) and (20). The following (measured) parameter values were used throughout the calculations below:

$C = 2.18$ μF, $R_1 = 2191$ Ω, $R_2 = 10{,}023$ Ω, $V_{IL} = 1.2730$ V, $V_{IH} = 1.2812$ V, $V_{DD} = 5.0280$ V

In order to find the maximum discharging time in (8), we set $A_{in} = V_{DD}$. That produces

$$t_{d,\max} = 9.44\,\text{ms} \tag{27}$$

In order to find the maximum charging time t_c, we set $A_{in} = 0$ in (20):

$$t_{c,\max} = 1.46\,\text{ms} \tag{28}$$

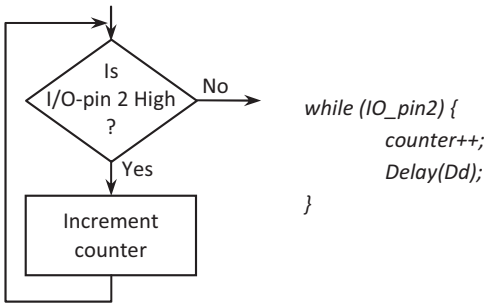

Fig. 13. Incrementing counter.

It follows that the starting value of N_0 should be

$$\frac{1.46}{1.46 + 9.44} \cdot 2^{12} = 548 \tag{29}$$

The microcontroller was a PIC18F458 from Microchip [22], clocked with an external crystal with nominal frequency $F_0 = 20.0000$ MHz. In this architecture, the internal instruction execution rate is $F_0/4$ and hence the instruction cycle period (ic) is $4/F_0$. If we first look at the part of the program that increments the counter variable (during discharging), see Fig. 13, the execution time of this loop must equal $t_{d,\max}$ (when $A_{in} = V_{DD}$).

The *counter++;* is executed in 1 ic (instruction cycle) only. The loop overhead produced by the used C-compiler (HI-TECH's PICC-18) consumed 8 ics. If the delay consumes D_d ics, the entire loop takes $9 + D_d$ ics to execute. When $A_{in} = V_{DD}$, this loop should be executed $2^{12} - N_0$ times in a time corresponding exactly to $t_{d,\max}$:

$$(2^{12} - N_0) \cdot (9 + D_d) \cdot \frac{4}{F_0} = t_{d,\max} \tag{30}$$

$$t_{d,\max} \Rightarrow D_d = \frac{F_0 \cdot t_{d,\max}}{4 \cdot (2^{12} - N_0)} - 9 \tag{31}$$

We may consider this to be another design rule; F_0, $t_{d,\max}$ and N_0 must be chosen so that $D_d \geq 0$ (preferably =0 to speed up the sampling rate). If D_d is a small number, it can be implemented as a number of "no operation" (nop) assembler instructions (asm("nop")). With $t_{d,\max} = 9.44$ ms, $N_0 = 548$ and $F_0 = 20$ MHz, we get $D_d = 4$.

The part of the program that decrements the counter variable (during charging), see Fig. 14, is treated correspondingly; the counter variable should be decremented exactly N_0 times during $t_{c,\max}$ (when $A_{in} = 0$).

$$N_0 \cdot (9 + D_c) \cdot \frac{4}{F_0} = t_{c,\max} \tag{32}$$

$$t_{c,\max} \Rightarrow D_c = \frac{F_0 \cdot t_{c,\max}}{4 \cdot N_0} - 9 \tag{33}$$

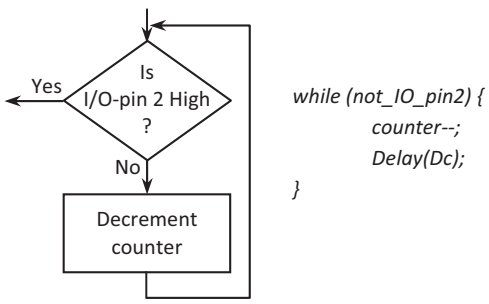

Fig. 14. Decrementing counter.

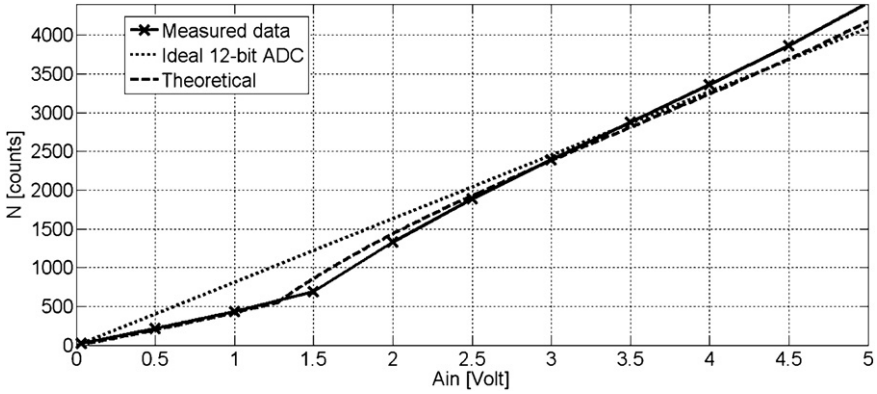

Fig. 15. The In–Out characteristics of the Analog-Direct circuit in Fig. 3.

This is yet another design rule; F_0, $t_{c,\max}$ and N_0 must be combined so that $D_c \geq 0$ (preferably = 0). With $t_{c,\max} = 1.46$ ms, $N_0 = 548$ and $F_0 = 20$ MHz, we get $D_c = 4$.

The last box in the firmware flowchart in Fig. 6, "ADC out = counter", could mean that we simply output the n-bit result in binary format on n LEDs (Light Emitting Diodes). However, in this application data was transferred to a host Windows computer via an asynchronous serial link at a baud rate of 9600. This routine consumed 28,774 ics. The time consumed by the main loop in Fig. 6 depends on A_{in}, i.e. how long it takes to charge/discharge the capacitor. The worst case, when $A_{in} = V_{DD}$, will determine the system's sample rate. In the worst case, the counter has to be incremented from N_0 to $2^{12} - 1 = 4095$, i.e. executing the loop in Fig. 13 takes exactly $t_{d,\max}$. The rest of the firmware in the main loop in Fig. 6, including initializing the counter, configuring the I/O-pins, loop overhead, etc., takes 774 ics. In total, the worst case takes

$$T_S = (28,774 + 774)ics + t_{d,\max} \tag{34}$$

to execute. If $t_{d,\max} = 9.44$ ms and the ic period = 4/(20 MHz) = 200 ns, then $T_S = 15.3$ ms. This corresponds to the minimal sampling interval, indicating a maximum sampling rate of 65 S/s. This should be compared to the sampling time of 17.5 ms achieved by the ΣΔ ADC implementation suggested by Peter et al. [5]. This should also be compared to the system's input bandwidth. Notice in Fig. 3 that during the sampling phase (when the capacitor is charged by A_{in}), I/O-pin 1 and 2 are High-Z inputs which means that R_2 and C will serve as a first order low-pass filter. This works greatly to our advantage since it will cancel random noise in A_{in} and serve as an anti-aliasing filter. That also gives us another design rule. The bandwidth of this anti-aliasing filter should be less than $1/2\ T_S$:

$$\frac{1}{2\pi R_2 C} \leq \frac{1}{2T_S} \Rightarrow R_2 C \geq \frac{T_S}{\pi} \tag{35}$$

We have $R_2 = 10{,}023\ \Omega$ and $C = 2.18\ \mu$F:

$$10023 \cdot 2.18 \cdot 10^{-6} = 0.022 > \frac{15.3 \cdot 10^{-3}}{\pi} = 0.0048 \tag{36}$$

Finally, we also need to relate the counter variable value N to the charging/discharging times in expressions (8) and (20). Since we got both D_d and $D_c = 4$, both loops in Figs. 13 and 14 takes 9 + 4 = 13 ics to execute and if each ic = $4/F_0 = 200$ ns, each counter increment/decrement corresponds to $t_{loop} = 13 \times 200$ ns = 2.6 μs (measured to 2.6017 μs, see Section 4). When $A_{in} \geq V_{IH}$, the counter is incremented from N_0 and if $A_{in} < V_{IH}$, the counter is decremented from N_0. Hence, we assign a time to each A_{in} according to the following expressions:

$$t_c = (N_0 - N_c) \cdot t_{loop} \quad \text{if} \quad A_{in} < V_{IH} \tag{37}$$

$$t_d = (N_d - N_0) \cdot t_{loop} \quad \text{if} \quad A_{in} \geq V_{IH} \tag{38}$$

where t_c and t_d are the expressions (20) and (8), respectively, and N_c and N_d represents the firmware counter variable value N during charging and discharging, respectively. This relates the counter value to the analog input voltage A_{in}:

$$N_c = N_0 - \frac{t_c}{t_{loop}} = N_0 + \frac{1}{(t_{loop}/C)((1/R_1)+(1/R_2))} \cdot \ln\left\{\frac{V_{IH}(R_1+R_2) - R_2 V_{DD} - R_1 A_{in}}{R_2(A_{in} - V_{DD})}\right\} \quad \text{if} \quad A_{in} < V_{IH} \tag{39}$$

$$N_d = \frac{t_d}{t_{loop}} + N_0 = N_0 - \frac{1}{(t_{loop}/C)((1/R_1)+(1/R_2))} \cdot \ln\left\{\frac{1}{R_2}\left(\frac{V_{IL}(R_1+R_2)}{A_{in}} - R_1\right)\right\} \quad \text{if} \quad A_{in} \geq V_{IH} \tag{40}$$

In Fig. 15 we have plotted N versus A_{in} according to the theoretical predictions in (39) and (40). In Fig. 15 we have also plotted experimentally registered calibration data and from Fig. 15 it is clear that the experimental data confirms the theoretical predictions in (39) and (40).

4. Methods and materials

Resistors and capacitor values (R_1, R_2, C) for our design example presented in Fig. 15, were measured with a HP4261A LCR meter. V_{DD} was measured with a digital desktop multi meter Agilent 34401A, to 5.0280 V. In order to register calibration data, a digital function generator (Agilent 33220A) was used as a signal source for A_{in}. This function generator can be configured for DC output and the advantage of using this function generator to simulate A_{in} instead of a general purpose DC power supply is that the output impedance is well-known (and "low", 50 Ω nominal).

The microcontroller used was a PIC18F458 from Microchip [22] and the parameters R_{in}, R_{OL}, R_{OH}, V_{IH} and V_{iIL}, were measured as described below (most of these methods were suggested by Reverter et al. [23]).

Fig. 16 illustrates how R_{in} and R_{OL} were measured.

An Agilent 34401 DMM (digital multi meter) was used to measure U_X. R_p was adjusted until $U_X = V_{DD}/2$ at which $R_p = R_{in}$ (or R_{OL}, respectively). Then R_p was measured with the HP4261A LCR meter. Using this method, we got $R_{in1} = R_{in2} = 1.012$ MΩ, and $R_{OL} = 21.79\ \Omega$. (Normally, the input impedance of the 34401A DMM is 10 MΩ but in order to measure R_{in} we reconfigured the input impedance to ">10 GΩ" so that it would not interfere with the R_{in}-measurement.)

R_{OH} was measured in a similar way as illustrated in Fig. 17.

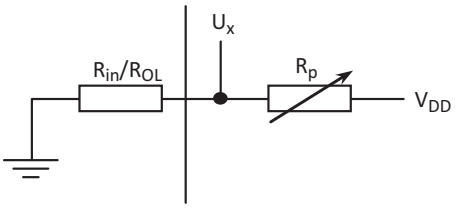

Fig. 16. Measuring R_{in}/R_{OL}.

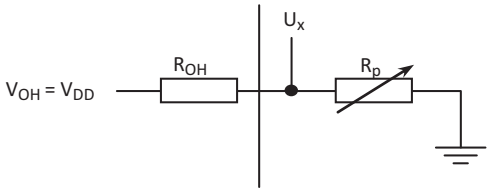

Fig. 17. Measuring R_{OH}.

First we verified that V_{OH} was equal to V_{DD} by measuring the "open output" voltage of the I/O-pin when set high with no external load applied. Then we connected the potentiometer and adjusted R_p again until $U_X = V_{DD}/2$ at which $R_p = R_{OH}$. This produced $R_{OH} = 57.15\ \Omega$.

The input logic high and low thresholds of I/O-pin 2 (V_{IH} and V_{IL}) were measured as illustrated in Fig. 18.

A small test firmware was written that configured I/O-pin 1 as input and I/O-pin 2 as output and then just executed a single line of code in an infinite loop: *I/O_pin2 = I/O_pin1;*. U_X was measured with the Agilent 34401A DMM. By slowly increasing the potential on I/O-pin 1 (by adjusting the potentiometer) until the LED was turned on, V_{IH} could be determined (=1.2812 V). Reversing this process, i.e. slowly decreasing the potential U_X from V_{DD} until the LED is turned off, produced $V_{IL} = 1.2730$ V.

Fig. 19 illustrates the hardware details of the proposed direct analog-to-microcontroller interface.

The FTDI-chip in Fig. 19 is a TTL-to-USB converter [24] that was used to convert the TTL digital output from the microcontroller's asynchronous serial interface to USB-formatted data in order to transfer data to the host Windows computer via an USB-interface. A simple terminal program was used to receive and store all data. Data was later analyzed in MATLAB.

Finally, t_{loop} was measured by first measuring the ic period (instruction cycle period). A small test program toggled an I/O-pin on each ic and by measuring the positive width of this signal we found that ic = 200.131 ns with a standard deviation of 0.073 ns. This was measured by using the embedded measurement tools of a Tektronix digital oscilloscope, TDS5032B. This indicates that $t_{loop} = 2.6017\ \mu$s with a standard deviation of 0.95 ns. The number of instruction cycles actually executed in each loop was verified in the simulator MPSIM in Microchip's integrated development environment (MPLAB [25]).

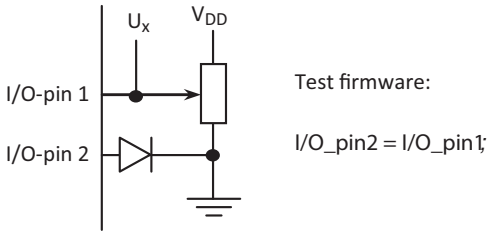

Fig. 18. Measuring V_{IH} and V_{IL}.

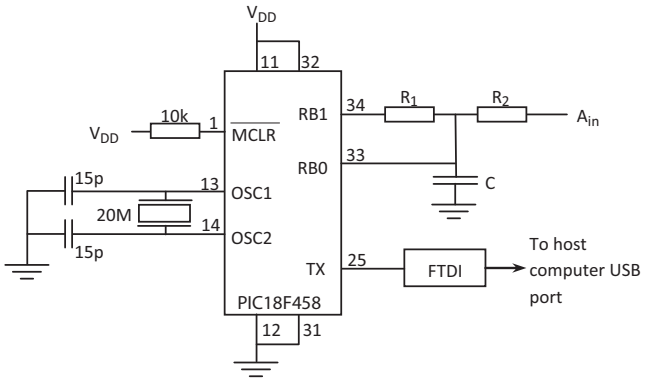

Fig. 19. Hardware details.

5. Uncertainty analysis

Expressions (21) and (9) relate the charging and discharging times, respectively, to the input analog voltage and expressions (37) and (38) relate the charging/discharging times to the firmware counter variable N. Inserting (37) into (21) and (38) into (9), we get two estimators for the unknown input analog voltage A_{in}:

$$A_{inL} = (V_{IH}(R_1 + R_2) - R_2 \cdot V_{DD}(1 - e^{-(1/C)((1/R_1)+(1/R_2))\cdot(N_0-N)\cdot t_{loop}})) \cdot (R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot(N_0-N)\cdot t_{loop}})^{-1} \quad \text{if} \quad A_{in} < V_{IH} \tag{41}$$

$$A_{inH} = V_{IL} \cdot (R_1 + R_2) \cdot (R_1 + R_2 \cdot e^{-(1/C)((1/R_1)+(1/R_2))\cdot(N-N_0)\cdot t_{loop}})^{-1} \quad \text{if} \quad A_{in} \geq V_{IH} \tag{42}$$

where we use A_{inL} and A_{inH} to distinguish between the two expressions for A_{in} depending on whether A_{in} is greater than V_{IH} or not. From these estimators we can see that

$$A_{inL} = f(V_{IH}, R_1, R_2, V_{DD}, C, N, t_{loop}) \tag{43}$$

$$A_{inH} = f(V_{IL}, R_1, R_2, C, N, t_{loop}) \tag{44}$$

(N_0 is not included since it is a firmware constant with zero uncertainty). In order to find the uncertainty in our estimation of A_{in} we need to know how uncertainties in any of the variables determining A_{in}, propagate into uncertainties in A_{in}. When doing that analysis, we will assume all variables to be independent (or at least uncorrelated). This is indeed true, with the exception of V_{IH} and V_{DD} in expression (43); V_{IH} most likely depends on V_{DD}, but we deliberately disregard that fact in the following calculations, for the following reasons: (1) the correlations coefficient between V_{IH} and V_{DD} is not known and (2) it would make calculations overwhelming. We are aware of the shortcomings of this approximation and we will keep it in mind when we analyze the results. We will illustrate by two examples how the uncertainty in A_{in} is calculated; the first example is for $A_{in} \geq V_{IH}$ (and we use expression (42)) and the other example is for $A_{in} < V_{IH}$ (and we use expression (41)).

Assume that X_i is a stochastic variable with standard uncertainty $u(x_i)$, and that another stochastic variable Y depends on X_i:

$$Y = f(X_1, X_2, X_3, \ldots) \tag{45}$$

We define the *sensitivity coefficient* as [26,27]:

$$c_i = \left.\frac{\partial f}{\partial X_i}\right|_{X_i = \widehat{X}_i} \tag{46}$$

Table 1
Uncertainty budget, $N \geq N_0$.

Parameter	Estimated/measured value	Standard uncertainty $u(x_i)$	Sensitivity coefficient $\lvert c_i \rvert$	$c_i \cdot u(x_i)$
V_{IL}	1.2730 V	0.0289 mV	1.57	4.54×10^{-5} V
R_1	2911 Ω	0.289 Ω	4.08×10^{-4} A	1.18×10^{-4} V
R_2	10,023 Ω	0.289 Ω	5.38×10^{-6} A	1.56×10^{-6} V
C	2.18 μf	2.89 nF	3.85×10^{5} V F^{-1}	1.11×10^{-3} V
N	1428.45	0.73	9.53×10^{-4} V	6.96×10^{-4} V
t_{loop}	2.6017 μs	0.95 ns	3.22×10^{5} V s^{-1}	3.06×10^{-4} V
A_{in}	1.998993 V	–	–	1.35×10^{-3} V

Table 2
Uncertainty budget, $N < N_0$.

Parameter	Estimated/measured value	Standard uncertainty $u(x_i)$	Sensitivity coefficient $\lvert c_i \rvert$	$c_i \cdot u(x_i)$
V_{IH}	1.2812 V	0.0289 mV	1.15	3.32×10^{-5} V
R_1	2911 Ω	0.289 Ω	7.84×10^{-5} A	2.27×10^{-5} V
R_2	10,023 Ω	0.289 Ω	1.63×10^{-5} A	4.72×10^{-6} V
V_{DD}	5.0280 V	0.0289 mV	1.34×10^{-1}	3.89×10^{-6} V
C	2.18 μf	2.89 nF	2.18×10^{5} V F^{-1}	6.30×10^{-4} V
N	286.59	0.91	1.81×10^{-3} V	4.52×10^{-4} V
t_{loop}	2.6017 μs	0.95 ns	1.81×10^{5} V s^{-1}	1.72×10^{-4} V
A_{in}	0.71776 V	–	–	7.95×10^{-4} V

If all variables X_i are uncorrelated, then the uncertainty in Y is

$$u(y) = \sqrt{\sum_{all\, i} (c_i \cdot u(x_i))^2} \tag{47}$$

So, in order to find the uncertainty of A_{in} we must find out how the uncertainty $u(x_i)$ of each variable propagates into an uncertainty $c_i \cdot u(x_i)$ in A_{in} and then take the square root of the square sum of all these uncertainties as in (47). First we must find the uncertainties of all parameters in expressions (41) and (42). The parameters that were measured with a digital instrument are treated as uniformly distributed stochastic variables with distribution limits equal to ±0.5 of the "weight" of the least significant digit. Hence, the width of the distribution function corresponds to the "weight" of the least significant digit and the standard deviation of a uniform distribution with a width of 2δ is $\delta/\sqrt{3}$ [26,27]. That is how we found the uncertainties of all digitally measured parameters in (41)–(42) which included all parameters except N and t_{loop}. N is the firmware variable and was estimated as the arithmetic average of 20 samples. The standard deviation of this data was estimated as follows:

$$\hat{\sigma}_N = \sqrt{\frac{1}{n-1} \sum_i (N_i - \bar{N})^2} \tag{48}$$

(where $n = 20$ is the number of samples) and the standard deviation of the estimated arithmetic average $\bar{N}$ is

$$\hat{\sigma}_{\bar{N}} = \frac{\hat{\sigma}_N}{\sqrt{n}} \tag{49}$$

t_{loop} was found in a similar way by recording the loop time (as described in Section 4) 10 times.

All these uncertainties are presented in the uncertainty budget in Table 1 [26,27] for a case where $A_{in} \geq V_{IH}$. In order to find the sensitivity coefficients in (46), we need to differentiate A_{in} with respect to each one of the parameters in (42). We used the WolframAlpha computational engine to do that. The results are presented in the uncertainty budget in Table 1.

The uncertainty budget in Table 1 represents a measurement of some unknown analog input voltage $A_{in} \geq V_{IH}$ which produced $\hat{N} = \bar{N} = 1428.45$. Since $N > N_0$, we use expression (42) to estimate A_{in}. From Table 1 we can see that this produces $\hat{A}_{in} = 1.99899$ V with a standard uncertainty of 0.00135 V. From Table 1 we can also see that in order to improve the precision, we should first of all try to get a more precise value of the C parameter (as expected since it is the parameter value determined with the poorest precision). The measurement uncertainty should be compared to the inherent quantization uncertainty of a 12-bit ADC which is (5 V)/4096 = 0.00122 V; the $1-\sigma$ uncertainty corresponds to 1.1 LSB.

Table 2 illustrates an example of an uncertainty analysis for the case where $A_{in} < V_{IH}$ and expression (41) is used. In this case the analog input voltage was estimated to 0.71776 ± 0.00080 V (where the uncertainty interval corresponds to one standard uncertainty; approximately 0.66 LSB). From Table 2 we can also see that our assumption about V_{IH} and V_{DD} being uncorrelated is of no major importance since the contributions of V_{IH} and V_{DD} are small compared to other contributions. In a correct analysis, we should also include the covariance between V_{IH} and V_{DD}, but that will not make any noticeable difference in the end result (as long the precision of the other parameters aren't improved first).

The uncertainty budgets in Tables 1 and 2 are examples of how to calculate the measurement uncertainty for all measured values in Fig. 15 and the maximum observed uncertainty corresponds to 1.15 LSBs. This occurs for the case when $N > N_0$ and maximum stochastic fluctuations in N.

6. Conclusions

This work has presented a simple "direct" interface technique for analog (DC) voltage signals to a microcontroller. The solution requires only two digital I/O-pins, two resistors and a capacitor. A design example has been analyzed in detail; it has a maximum sampling rate of 65 S/s and a typical precision that corresponds to (or outperforms) the quantization uncertainty of a 12-bit ADC. The solution is a mixture of the "Direct Sensor-to-Controller" technique for resistive and capacitive sensors suggested in [1–4,11–18] and the ΣΔ ADC implementations suggested in [5–10]. This solution does not need any internal or external building blocks at all (as the ΣΔ ADC implementation does). The disadvantage of the proposed circuit solution is of course its limited bandwidth.

Compared to the ΣΔ ADC solution, this solution does not have the noise shaping properties of a ΣΔ ADC [19], but is on the other hand implementable in the simplest possible digital system since only two I/O-ports are required; the ΣΔ ADC solution requires an integrated comparator. The fact that no comparator is required indicates that also inherently digital systems like CPLDs and FPGAs

can take advantage of this solution. The fact that I/O-pins must have tri-state capability is not a problem; CPLDs/FPGAs programmed in VHDL can have tri-state I/O-pins if they are defined as "inouts" in the entity and declared as std_logic types ('Z' = HighZ).

This work has also presented a thorough theoretical analysis of the proposed system and demonstrated that experimental data agrees very well with theoretical predictions. We have also demonstrated how uncertainties in input parameters propagate to an uncertainty in the output parameter.

As predicted by expressions (41) and (42) the relationship between the analog input signal and the measured integer N is not linear. However, this is not a major problem and is easily handled by implementing a LUT (Look-Up Table) in firmware; when applied to non-linear sensors like thermo couples, LUTs must be implemented anyway.

References

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

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

[3] C.B. Baker, Building a 10-bit Bridge Sensing Circuit using the PIC16C6XX and MCP601 Operational Amplifier, Microchip Technology Inc., Chandler, Arizona, 1999, Application Note AN717.

[4] L. Bierl, Precise Measurements with the MSP430, Application Report, Texas Instruments, 1996.

[5] D. Peter, B.C. Baker, D. Butler, Make a Delta-Sigma Converter Using a Microcontroller's Analog Comparator Module, Microchip Technology Inc., Chandler, Arizona, 1998, Application Note AN700.

[6] J.D.B. Soldera, M. Espindola, Olmos, Implementing a 10-bit Sigma-Delta Analog-to-Digital Converter Using the HC9S08Rx MCU Family Analog Comparator, Freescale Semiconductor, 2005, Application Note AN2688, Rev. 0.1.

[7] STMicroelectronics, Implementation of sigma-delta ADC with ST7FLUE05/09, in: STMicroelectronics Application Note 1827 [online] (updated 29 February 2008), 2008, Available at: http://www.st.com/stonline/books/pdf/docs/10304.pdf (accessed 28 April 2011).

[8] P. Weber, C. Windish, Build a complete industrial-ADC interface using a microcontroller and a sigma-delta modulator, in: Electronic Design Strategy News, July 5, 2007 [online] (updated August 1, 2007), pp. 63–66, 2007, Available at: http://www.edn-europe.com/buildacompleteindustrialadcinterfaceusingamicrocontrollerandasigmadeltamodulator+article+1668+Europe.html (accessed May 5, 2011).

[9] Unknown, Integrated ADC for Altera Cyclone-IV Devices, Missing Link Electronics, Technical Brief 20110419 [online] (updated September 8, 2011), 1997, Available at: http://www.missinglinkelectronics.com/files/papers/MLE-TB20110426.pdf (accessed September 30, 2011).

[10] F. Harris, C. Dick, FPGA Signal Processing Using Sigma-Delta Modulation [online] (updated November 27, 2007), 1999, Available at: http://www.signumconcepts.com/download/paper009.pdf (accessed September 30, 2011).

[11] F. Reverter, M. Gasulla, R. Pallàs-Areny, A low-cost microcontroller interface for low-value capacitive sensors, in: IMTC 2004 – Instrumentation and Measurement Technology Conference, Como, Italy, May 18–20, 2004.

[12] F. Reverter, Ò. Casas, Direct interface circuit for capacitive humidity sensors, Sensors and Actuators A 142 (2008) 315–322.

[13] J.E. Gaitán, M. Gasulla, R. Pallàs-Areny, Analysis of a direct interface circuit for capacitive sensors, IEEE Transactions on Instrumentation and Measurements 58 (September (9)) (2009) 2931–2937.

[14] F. Reverter, Ò. Casas, A microcontroller-based interface circuit for lossy capacity sensors, Measurement Science and Technology 21 (2010) 065203.

[15] F. Reverter, Ò. Casas, Interfacing differential capacitive sensors to microcontrollers: a direct approach, IEEE Transactions on Instrumentation and Measurement 59 (October (10)) (2010) 2763–2769.

[16] E. Sifuentes, Ò. Casas, F. Reverter, R. Pallàs-Areny, Direct interface circuit to linearise resistive sensor bridges, Sensors and Actuators A 147 (2008) 210–215.

[17] J. Jordana, R. Pallàs-Areny, A simple, efficient interface circuit fro piezoresistive pressure sensors, Sensors and Actuators A 127 (2006) 69–73.

[18] A. Custidio, R. Bragós, R. Pallàs-Areny, A novel sensor-brigde-to-microcontroller interface, in: IEEE Instrumentation and Measurement Technology Conference, Budapest, Hungary, May 21–23, 2001.

[19] M.W. Hauser, Principles of oversampling A/D conversion, Journal of Audio Engineering Society 39 (January–February (1 (2))) (1991).

[20] D. Jarman, A Brief Introduction to Sigma Delta Conversion, Intersil Application Note AN9504, May 1995 [online] (updated November 1, 2002), 1995, Available at: http://www.intersil.com/data/an/an9504.pdf (accessed April 21 2011).

[21] W. Kester, The Data Conversion Handbook, Analog Devices Inc., ISBN: 0-916550-27-3 [online] (updated May 1, 2007), 2004, Available at: http.//www.analog.com/library/analogDialogue/archives/39-06/data_conversion_handbook.html (accessed September 9, 2011).

[22] Microchip, PIC18FXX8 Data Sheet, Microchip Inc., DS41159C, Tuscon, Arizona [online] (updated: dynamic), 2003, Available at: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en 010301 (accessed Mars 12, 2011).

[23] F. Reverter, J. Jordana, M. Gasulla, R. Pallàs-Areny, Accuracy and resolution of direct resistive sensor-to-microcontroller interfaces, Elsevier: Sensors and Actuators A 121 (2005) 78–87.

[24] Future Technology Devices International, TTL to USB Serial Converter Generic Cables [online] (updated December 7, 2011), 2011, Available at: http://www.ftdichip.com/Support/Documents/DataSheets/Cables/DS_TTL-232RG_CABLES.pdf (accessed January 15, 2012).

[25] Microchip, MPSIM User's Guide [online] (updated March 26, 2006), 2006, Available at: http://ww1.microchip.com/downloads/en/devicedoc/30027i.pdf (accessed January 18, 2011).

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

[27] S. Bell, A Beginner's Guide to Uncertainty of Measurement [online] (updated July 13, 2009), 2009, Available at: http://www.wmo.int/pages/prog/gcos/documents/gruanmanuals/UK_NPL/mgpg11.pdf (accessed December 25, 2011).

Biography

Lars Bengtsson is currently a senior lecturer in embedded systems at the University of Gothenburg in Sweden (since 2007) and was prior to 2007 a senior lecturer at Chalmers University of Technology. His area of research is embedded measurement systems and he is the author of several university textbooks on embedded systems and electrical measurement techniques.