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 $\Sigma\Delta$ 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 $\Sigma\Delta$ 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 $\Sigma\Delta$ 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

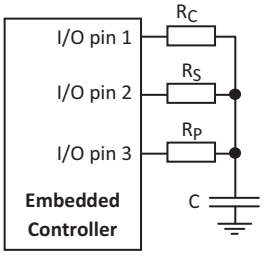

Fig. 1. The one-point calibration technique.

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

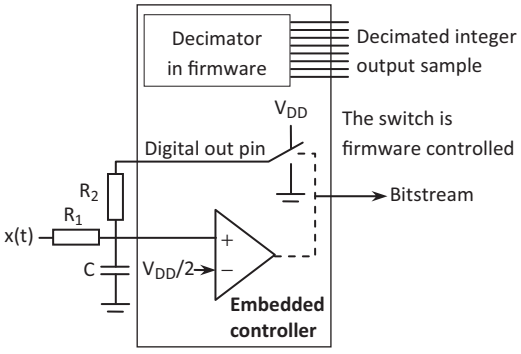

Fig. 2. Implementing a ΣΔ ADC in an embedded controller.

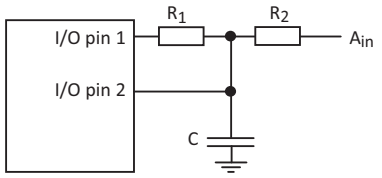

Fig. 3. Analog-to-digital converter.

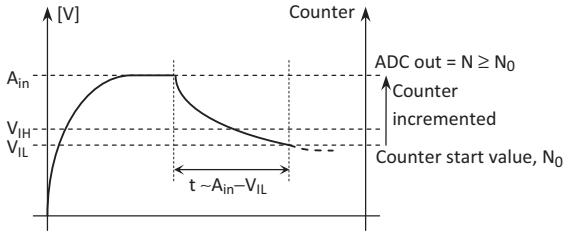

Fig. 4. Case 1: $A_{in} \geq V_{IH}$.

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.

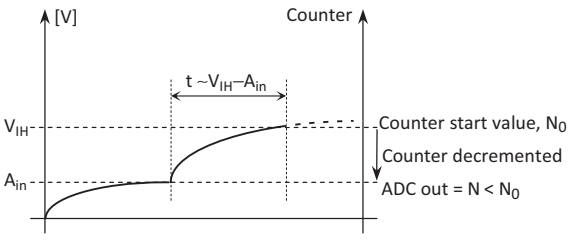

Fig. 5. Case 2: $A_{in} < V_{IH}$.