Chapter 10
The Analog Signal Chain

Embedded systems interact with the outer real world, processing information received from it and delivering back an output to modify the performance of a device or to provide information needed to make decisions. In this environment, most signals to be processed are analog. Temperature is not simply hot or cold, pressure high or low, light intensity bright or dark. They all fall within a wide range of possible values. Hence, it becomes necessary to interface this analog world with our digital embedded systems.

Signal processing may be digital, analog, or a combination of both worlds, i.e., mixed signal. In fact, the "MSP" part of the microcontroller name MSP430 emphasizes this focus, "Mixed Signal Processor". Hence, several of the peripherals embedded in different models belong to the analog portion of the signal processing, the analog signal chain.

10.1 Introduction to Analog Processing

The term *signal chain*, short for signal-processing chain, refers to a series of signal-conditioning electronic components connected in tandem, or cascade. That is, the output of one portion of the chain supplies input to the next one. In mixed-signal systems, the term *analog signal chain*, or simply *analog chain*, focuses on the components that process the analog signals prior to the digital processor, or that extract analog information from the digital processor's output. The complete process is illustrated in Fig. 10.1.

Inputs to the system may be temperature, pressure, humidity, speed, flow rate and so on. Most often, the first block in the system transforms the information into an electrical signal such as voltage or current. This operation is known as *sensing*, and is based on *sensors*. These are components that receive as stimulus the physical magnitude of interest, or a change in that magnitude, and deliver a value or a modulation in an electrical signal related to the input. This electrical magnitude may be a voltage, current, resistance, charge, or an electrical or magnetic field.

DOI: 10.1007/978-1-4614-3143-5_10,

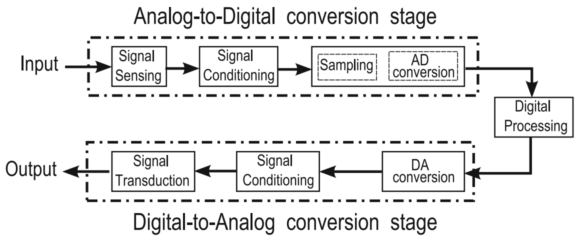


Fig. 10.1 Block description for mixed-signal processing chain

Often, the electrical signal provided by the sensing stage is not adequate for direct use in the chain, and should be processed with a *conditioning circuit*. For example, the sensing of pressure may produce a voltage in the millivolt range, insufficient for processing. Therefore, this signal should be amplified to bring it up to an appropriate volt range. Amplification is very common in conditioning. Other situations may require filtering, linearization, impedance conversion, and so on.

The conditioning stage output is fed into an *analog-to-digital converter* (*ADC*) stage whose output is then used as input to the digital microcontroller or processor. There are different ADCs and this component may include subsystems such as the sample-and-hold portion.

The ADC's output are digital words fed to the digital processor where they are utilized according to the user's program. This digital processor in turn will deliver n-bit words as output. Depending on the application, these words may be directly used (perhaps via a conditioning or interfacing circuit) or may need to be translated into an analog value, initiating a digital-to-analog conversion phase. The first step in this process is the *digital-to-analog converter* (DAC) which will deliver a voltage or current output whose value is proportional to the decimal equivalent of the digital input word. As in the previous phase, this analog output may need further conditioning such as filtering, smoothing and so on. The final step may be the transducer, which operates akin to the sensor, transforming an electrical signal energy into another type of energy. For example, the speaker translates current or voltage variations into acoustic information.

Important electronic components used in the analog-signal-chain are sensors, operational amplifiers, comparators, operational amplifiers, analog-to-digital and digital-to-analog converters, among others. In addition, supporting the operation of the main players we have reference voltage and reference current, LDO, sample-and-hold circuits, and so on. A brief introduction to some of the components in the analog chain devices is presented in the following sections.

10.2 Analog Signal Chain in MSP430

The MSP430 family of microcontrollers was projected as a mixed signal device. Therefore, several of the devices needed in the analog chain have been included in several models. Table 10.1 summarizes the available analog peripherals in MSP430

Table 10.1 Analog signal chain devices in MSP430

Series	Comparator	OA	DAC	ADC	REF	LDO
’x3	No	No	No	Yes	No	No
’x1	Yes	No	Yes	Yes	No	No
’x2	Yes	Yes	Yes[a]	Yes	No	No
’x4	Yes	Yes	Yes[a]	Yes	No	No
’x5&’x6	Yes	No	Yes[a]	Yes	Yes	Yes

[a] SAR and Sigma-Delta converters

controllers for some models in the various families. The devices can be configured to receive external or internal inputs, and also to route their output to external terminals. The reader should consult the user guides and data sheets for full details on specific family members.

10.3 Sensors and Signal Conditioning

Many system applications require the measurement or detection of a physical or chemical or electrical quantity or condition. Detectors are often used as on/off switches to signal the presence of an environmental condition, even if the sensor function itself may be of analog type. Analog sensors are used to indicate the magnitude or change in this environmental condition by means of a value or change in an electrical property. Usual electrical magnitudes include voltage, current, resistance, capacitance, Hall-effect, and charge.

Usually, sensors are named after the application they are intended for, and not by the physical way in which they operate. Thus, we speak for example of proximity sensors. Also, many sensors are in fact integrated circuits, or MEMS systems that may or may not include conditioning circuits in the chip. Moreover, sometimes the term *transductor* is used as synonymous to sensor. But many authors and applied engineers use the term to indicate conversion from electrical energy to other type of energy, as for example in the case of speakers that translate current or voltage variations into acoustic waves. In this way, transductors are used in the output stage of the system. To finish this short discussions about terminology, engineers in other fields, like mechanical engineers, use “sensor” to speak of elements delivering a mechanical type energy.

10.3.1 Active and Passive Sensors

Sensors can be identified as active or passive. Active sensors generate a signal, normally current or voltage, in response to the environmental stimulation they are used for. Examples of such sensors are thermocouples and piezoelectric accelerometers. Most often the magnitude of the generated signal is quite small, requiring an amplification stage.

Passive sensors, on the other hand, produce changes in some passive quantity, like a resistance, or a capacitance, or an inductance. Some examples of passive sensors are the thermo-resistors or thermistors, capacitive humidity sensors, and others. Passive sensors require power sources to generate information. Often, the actual reading or processing is by means of voltage. For resistance type sensors, resistance sub-circuits like voltage/current dividers, Wheatstone bridge circuits or current sources are used, as illustrated in Fig. 10.2.

Capacitance is typically measured indirectly. For example, by using relaxation oscillator and measuring the charging time required to reach a threshold voltage, or by measuring the oscillator's frequency. A capacitive voltage divider using a fixed-frequency AC voltage signal is another measurement technique frequently used; more accurate instruments may use a capacitance bridge configuration.

10.3.2 Figures of Merit for Sensors

Sensors are not perfect. We need therefore ways to determine the fitness and validity for the intended applications. Some figures of merit associated to sensors are the following:

Sensitivity: This is mathematically defined by the derivative of output with respect to input, that is, the curve's slope. It represents the minimum input of physical parameter that will create a detectable standardized output change.

- *Sensitivity error*:Departure from the ideal slope.

Range: stated by the maximum and minimum values that can be measured.

- *Dynamic range* is the difference between the maximum and minimum measurable values. This range is also called *full scale*.

Precision: This concept is associated to the degree of measurement reproducibility. While an ideal sensor would output exactly the same value every time its input is subjected to the same magnitude of the measured parameter, real sensors

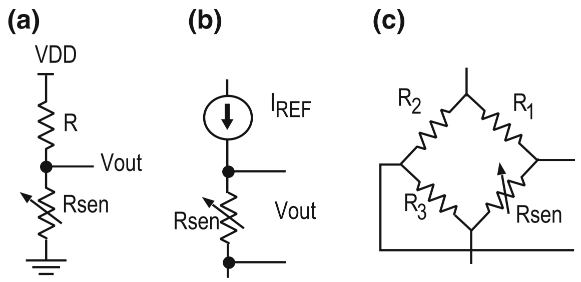


Fig. 10.2 Converting resistance to voltage information: **a** voltage divider; **b** with reference current; **c** with resistive bridge

outputs are distributed around a mean value. Precision is usually described in statistical terms.

Accuracy: The maximum difference that exhibited between the actual value and the value at the output of the sensor. True values are previously measured using a primary or good secondary standard. Accuracy can be expressed either as a percentage of full scale or in absolute terms.

Resolution: The smallest detectable incremental change of input parameter that can be detected in the output signal. It can be expressed either as a proportion of the reading (or the full-scale reading) or in absolute terms.

Offset: This parameter is defined as the output that will exist when it should be zero. In other words, it is a particular value of the accuracy error.

Linearity: The linearity, or more exactly *non-linearity*, refers to the extent to which the actual measured curve departs from an ideal linear curve determined by a least squares fit, line and the actual measured or calibration line. Linearity is often specified in terms of percentage of nonlinearity, as defined by (10.1).

- *Dynamic Linearity:*

Hysteresis: The property that tells how well the sensor or transducer follows changes of the input parameter regardless in which direction this occurs. Ideally the sensor should not have hysteresis.

Response Time: This is the time required for the output to change from its previous state to a final settled value within a tolerance band, $\pm\varepsilon$, of the correct new value. This becomes very important for rapid changes in the input parameter values.

Notice that precision and accuracy are not synonymous. A sensor or instrument can be very precise, yielding consistently similar inaccurate readings (Fig. 10.3a). Conversely, it can be acceptably accurate, with imprecise readings (Fig. 10.3b). Of course, it is desirable to have both precise and accurate devices (Fig. 10.3c).

An ideal sensor has a linear response to changes of the input magnitude. In fact, in many instances the conditioning circuit consist precisely in producing a linear curve. Most often, however, the actual response deviates from a linear behavior (Fig. 10.4), which is really a least squares fit. The maximum deviation determines the non-linearity factor with respect to full-scale as

$$Nonlinearity(\%) = \frac{\Delta_{in(max)}}{IN_{FS}} \times 100\,\% \tag{10.1}$$

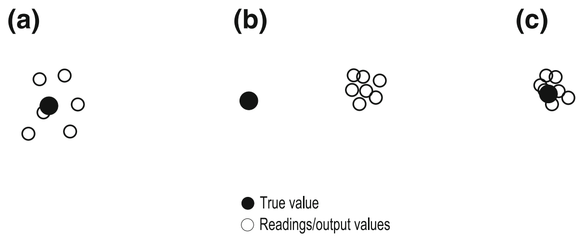


Fig. 10.3 Precision and accuracy: **a** "accurate" but imprecise readings; **b** precise but inaccurate readings; **c** precise and accurate readings

Fig. 10.4 Definition of linearity in terms of input

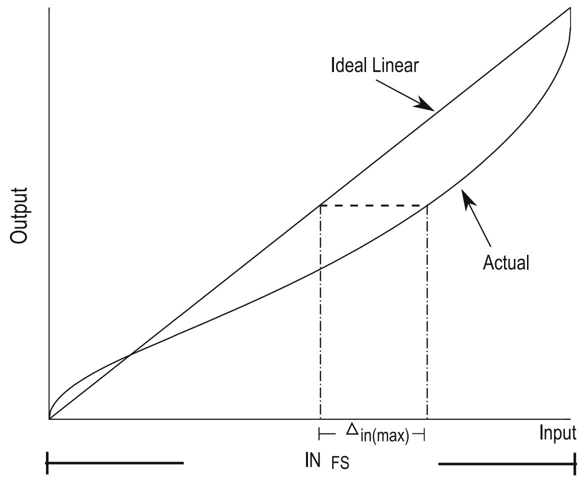
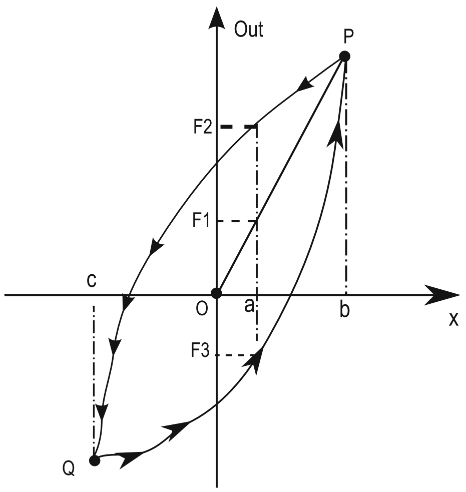


Fig. 10.5 Hysteresis curve



where $\Delta_{in(max)}$ is the maximum input difference required to obtain an ideal output equal to the actual output, and IN_{FS} is the full-scale input range.

The hysteresis information tells us how the sensor responds to direction of changes in the input values. Figure 10.5 illustrates this phenomenon. Starting from point O, when the input value x reaches the value "a", the output value will be F1. As long as the input value does not increase up to value "b", the output will be the same as before. However, the situation changes as soon as the input parameter gets to the value "b", and thus the curve to point P. When x decreases, it will follow the upper path, so for x = a the output will now be F2. when x goes down to x = c, reaching point Q, the path will now change for increasing values, and the output for x = a will now be F3. When the path reaches point P, the path will be again the upper one. Hence, we see that for input values between c and b, the output will depend on the path.

Example 10.1 *One of the most important magnitudes to measure is temperature. Even applications intended to measure other variables such as pressure, flow, or position, require temperature monitoring to ensure accuracy. The four most common temperature sensors are thermocouples, RTD (resistance-temperature-detectors), thermistors, and IC (integrated-circuit) sensors.*[1]

A thermocouple is built with two wires of dissimilar metals or alloys, with a weld bead bonding them on one end. When there is a temperature difference between the bead and the open end of the thermocouple wires, or this end is heated, a voltage difference, called Seebeck voltage, appears between the open end of the two wires. This voltage depends on temperature difference between wire ends. The major strengths of thermocouples are the wide temperature range of application and the durability. Linearity depends on the type, and in some applications the function must be approximated by at least a fourth order polynomial or a look-up table be used. Their sensitivity is in the range of tens of microvolts per degree Celsius with an accuracy of $\pm 0.5\,^{\circ}C$*. Practical use of thermocouples requires an isothermal block to compensate for parasitic thermocouples caused by the cooper connections.*

RTDs provide excellent accuracy in a temperature-sensing environment, and are a good choice for accurate temperature measurement. Their temperature range is narrower than that of thermocouples but wider than those of thermistors. RTD are made of metals, which have a positive temperature coefficient; that is, resistance increases with temperature. Platinum RTDs offer the most stable version and offer better linearity, with an accuracy within $\pm 4.3\,^{\circ}C$ *over its temperature range, and a sensitivity of* $0.00385\,\Omega/\Omega/^{\circ}C$*. For better accuracy, one may use a corrective equation or a look up table.*

Thermistors are by far the mostly widely used sensors in temperature applications. They range in price from 10 cents to approximately \$25, and find use in applications from automotive monitoring and exhaust-emissions control to ice detection, skin sensors, blood and urine analyzers, home appliances, mobile phones, and many more. Although thermistors are either NTC (negative temperature coefficient) or PTC (positive temperature coefficient), the former group is better for analog applications requiring precision temperature measurement, while the latter best suit switching applications. For computing purposes, one advantage of most NTC thermistors is that the curve can be approximated by the Steinhart-Hart equation

$$\frac{1}{T} = A + B \ln R + C(\ln R)^3 \tag{10.2}$$

where T *is the temperature in Kelvin,* R *is the thermistor resistance and* A*,* B *and* C *are curve-fitting coefficients found with three points from the R-T curve within a* $100\,^{\circ}C$ *range. The equation attains a* $\pm 0.02\,^{\circ}C$ *curve fit. A variation of this equation*

[1] Agilent's application note 290 at http://cp.literature.agilent.com/litweb/pdf/5965-7822E.pdf offers a very good introduction and tutorial on the use of these devices.

is an approximation given by

$$\ln R = a_o + a_1 \frac{1}{T} + a_3 \frac{1}{T^3} \tag{10.3}$$

which stems from the solution of the previous one. Usually the coefficients for either form are provided by the manufacturer.

The above equations for the thermistor or else a look up table can be used when using this sensor. Also, a rather good linearization technique can be achieved with an appropriate resistor in parallel.

The fourth group of temperature sensors are the Integrated-Circuit Temperature sensors. Their advantages include user-friendly output formats and easy installation during printed-circuit-board assembly. They have however slow response. They come in a variety of combinations of input and output. In analog form, the common output form are in current or voltage, with sensitivities of the order of 1 μA/K and 10 mV/K. Since they are already integrated, they can be combined with more circuitry and offer a digital output already. In addition, these sensors offer very good output linearity.

10.4 Operational Amplifiers

Operational amplifiers (OA) are one of the IC backbones for signal conditioning. The designer may select the adequate model based on the application needs and the figures of merit for the OA, such as the gain-bandwidth product, offset voltage and current, input and output impedances, S/N ratio, slew-rate, and CMRR, among others. Most OAs are compensated internally, but further compensation may be applied if necessary.

The voltage OA, or simply OA, whose symbol and transfer function are shown in Fig. 10.6, is basically a differential amplifier with a *non-inverting input* V_+ and an *inverting input* V_-. It is often biased using two DC power supplies VCC and VSS, usually VSS = −VCC. However, single power supply versions (i.e., VSS = 0 V) are also common and it is also possible to reduce the two source to one source version using appropriate circuiting. The power supply values, including ground for one supply cases, are called *rails*.

The output versus differential input voltage transfer curve is shown in Fig. 10.6b. As seen in the curve, for differential inputs large enough (in a positive or negative sense), the output voltage reaches a saturation value, which is always in the region $VCC \geq Vout \geq VSS$. Since the gain is very high, usually greater than 50,000, saturation is reached after several microvolts in difference. When the saturation values are equal to the rail values, the OA is said to be rail-to-rail OA.

Mathematically, in the linear region for a differential input $\varepsilon+ \geq (V_+ - V_-) \geq \varepsilon-$ the transfer curve can be expressed as $Ao(V_+ - V_-)$, where the differential gain Ao, called *open-loop gain*. This gain is very high, 80 dB or more, so the ε values are in the microvolt range. For this reason, numerical calculations use the *ideal OA* model,

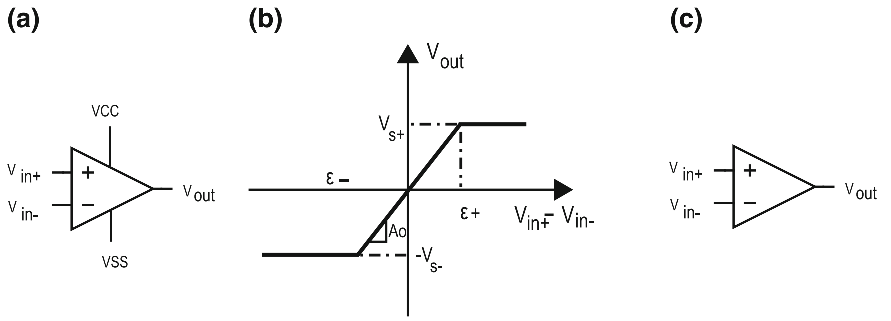


Fig. 10.6 Operational amplifiers: **a** symbol showing DC supplies, **b** transfer curve, **c** simplified common symbol

also called *infinite gain OA model* because it assumes $Ao = \infty$, defined as

$$V_{out} = \begin{cases} Vs_+ & \text{for } (V_+ > V_-) \geq \varepsilon+ \\ (V_+ - = V_-) & \text{for } Vs_+ \geq Vout \geq Vs_- \text{ (linear region)} \\ Vs_- & \text{for } (V_+ < V_-) \leq \varepsilon- \end{cases} \tag{10.4}$$

and

$$i_+ = i_- = 0 \tag{10.5}$$

Negative feedback is required for stability in analog applications in the linear region. Working with positive feedback or in open-loop condition keeps the output saturated. This condition is applied when using the OA as comparator.

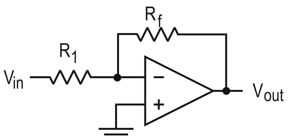
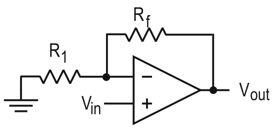
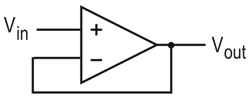
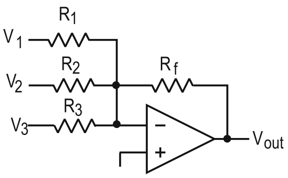
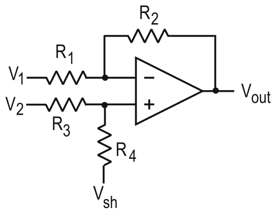
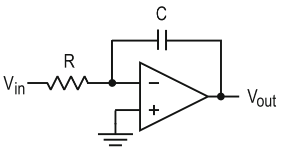
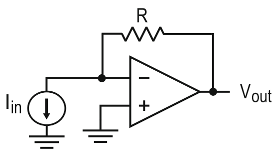
10.4.1 Basic OA Linear Configurations

Table 10.2 summarizes some important linear configurations used in analog applications, together with their *closed-loop gains* or formulas.

Resistance values in OA configurations cannot be very high because of associated stray capacitance problems. But they should be large enough to keep normal operating conditions on the OA without causing an excessive load to the signal. All configurations are valid for static inputs or slow signals. For fast changes, the inclusion of reactive elements may be required to deal with parasitics, as illustrated in the following example.

Example 10.2 *Figure 10.7 shows some applications of basic configurations for sensor interface. Inset (a) presents a connection for a resistive type sensor, like a thermistor. In this configuration,* Radj *is a potentiometer used for calibration with respect to a particular value. In thermistors, the resistance at* 25 °C *is a common reference.*

Table 10.2 Linear configurations using OA's

Configuration	Circuit	Formula
Inverting amplifier		$\frac{V_{out}}{V_{in}} = -\frac{R_f}{R_1}$
Non inverting amplifier		$\frac{V_{out}}{V_{in}} = 1 + \frac{R_f}{R_1}$
Buffer[a]		$\frac{V_{out}}{V_{in}} = 1$
Inverting adder		$V_{out} = -\left(\frac{R_f}{R_1}V_1 + \frac{R_f}{R_2}V_2 + \frac{R_f}{R_3}V_3\right)$
Differential amplifier[b] (w/offset)		$V_{out} = \frac{R_2}{R_1}(V_2 - V_1) + V_{sh}$ if $\frac{R_2}{R_1} = \frac{R_4}{R_3}$
Inverting integrator		$V_{out} = -\frac{1}{RC}\int V_{in}dt$
Current to voltage converter		$V_{out} = R\, I_{in}$

[a] Also known as voltage follower or unity gain
[b] Also known as difference amplifier; subtractor when $R_2R_3 \neq R_4R_1$

The second example in (b) shows the case for a current type sensor, using the current-to-voltage converter. Ideally, there is no current in the resistance Ro since this *is in parallel with the inputs of the OA. The voltage type sensor, as in the thermocouple case, can be interfaced with a voltage amplifier as shown in (c).*

An example of a current type sensor is the photodiode, which can be used in a current-to-voltage configuration. The voltage at the diode is practically zero, a condition which is known as photovoltaic mode. There are variants to this circuit, especially when using single supply OA's. When there are fast changes in input, the reactive parasitics of the resistor and the photodiode introduce frequency dependent components which degrade the performance considerably. Capacitor C is used to compensate for these problems by introducing a pole in the system.

Voltage V_{sh} in the difference, or differential, amplifier in Table 10.2 provides an offset or a shift to the output. The most general version presented in textbooks and in many applications has this terminal connected to ground. The offset is particularly useful for those single supply OA configurations to insure that the output falls within the applicable limits.

The formula for the differential depends on matching the ratios $R_2/R_1 = R_4/R_3$. For non-matching ratios, these amplifiers are also known as subtractor circuit with different weights for each voltage. Using nodal equations and the ideal OA model, it can be verified that

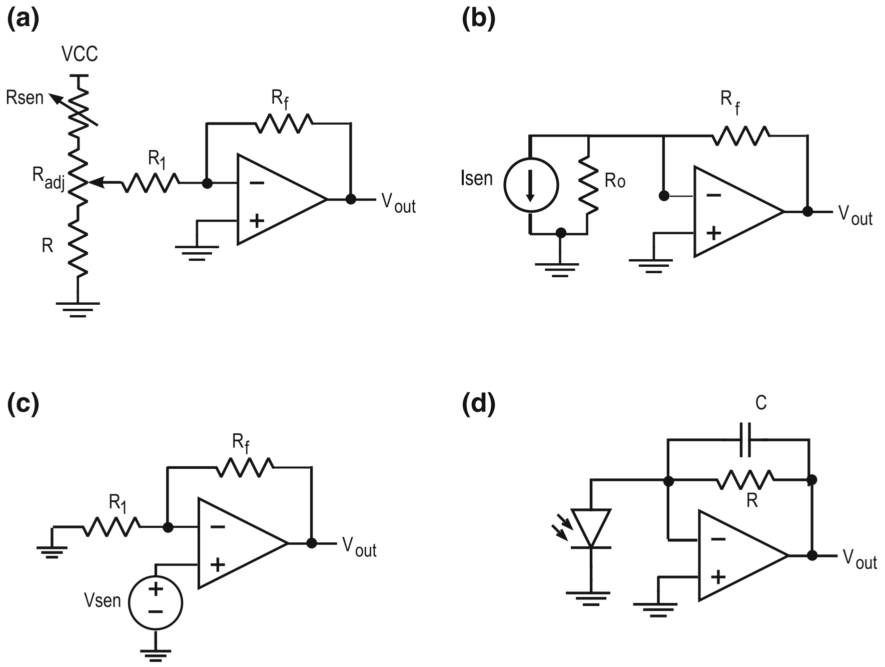


Fig. 10.7 Sensor conditioning examples with basic OA amplifiers: **a** for resistive type sensor, **b** a current type sensors; **c** voltage type sensor; **d** a photodiode (current type) with capacitive compensation for fast changes

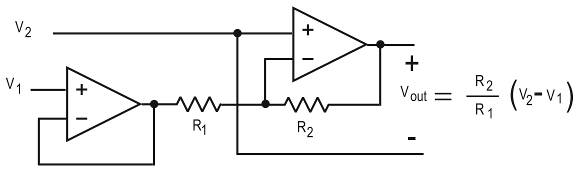


Fig. 10.8 Two OA differential floating output amplifier

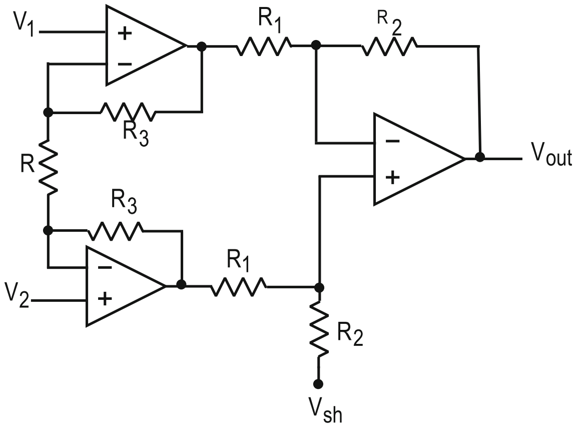


Fig. 10.9 Instrumentation amplifier (with output shifting V_{sh})

$$\frac{\frac{R_2}{R1}+1}{\frac{R_4}{R3}+1} = \frac{\frac{R_2}{R1}V_1 + V_{out}}{\frac{R_4}{R3}V_2 + V_{sh}} \tag{10.6}$$

This is the general case. If the resistance ratios are matched, we arrive at the formula given in the table.

Another differential amplifier built with two operational amplifiers and only two resistors is shown in Fig. 10.8. The output, however, is a floating one. This configuration is sometimes applied in embedded systems.

The loading effect on inputs in the difference amplifier is solved with the so called *instrumentation amplifier*. A simple version consists in the use of buffers at the inputs. A more general solution is presented in Fig. 10.9. This is also based on the previously introduced differential amplifier and offers additional gain control using resistance R. Instrumentation amplifiers are available in IC form from different vendors, with only the R resistance and shifting reference, when a terminal is available, being provided by user. In some cases, V_{sh} is grounded internally. For this instrumentation amplifier the output is given by

$$V_{out} = \left[1 + \frac{2\,R_3}{R}\right]\frac{R_2}{R_1}(V_2 - V_1) + V_{sh} \tag{10.7}$$

Example 10.3 *Differential and instrumentation amplifiers become very important in applications where for some reason or other we cannot use grounded signal* *information. The examples in Fig. 10.10 illustrate some configurations. Notice the use of the bridge configuration, where one or more of the resistances may vary with sensed magnitudes, depending on the applications.*

Although (a) and (b) look similar, the first case the sensor is a resistance, while in the second one it is the current source. In the former case, the current is kept constant using a current reference circuit. Any change in the amplifier input voltage is due to a variation in the resistance. On the other side, the circuit in (b) has a constant resistance, and changes in input voltage to the amplifier are due to the current response of the sensor to variations in the magnitude being monitored.

The bridge circuit in (c) can be driven either by a voltage supply or a reference current.

Important applications of OA fall in voltage-to-current conversions. One of the most popular circuits in this group is the *Howland current source*, of which two versions are shown in Fig. 10.11a, b.

It can be shown that for these circuits we have, respectively,

$$I_{out} = \frac{1}{R_3}(V2 - V1) \text{ if } \frac{R_2}{R_1} = \frac{R_4}{R_3} \tag{10.8}$$

for (a), and

$$I_{out} = \frac{R_2}{R_1 R_5}(V2 - V1) = \frac{1}{R_3}\left(1 + \frac{R_4}{R_5}\right)(V2 - V1) \text{ if } \frac{R_2}{R_1} = \frac{R_4 + R_5}{R_3} \tag{10.9}$$

for (b). Comparing the equations, it is seen that the improved version allows greater currents. The validity of these equations are limited by the output current and voltage limitations of the OA.

There are other voltage-to-current converters in the literature based on the original Howland source. Trimming is common in these circuits to achieve the matching conditions. OA limitations may force further compensation schemes; the reader can

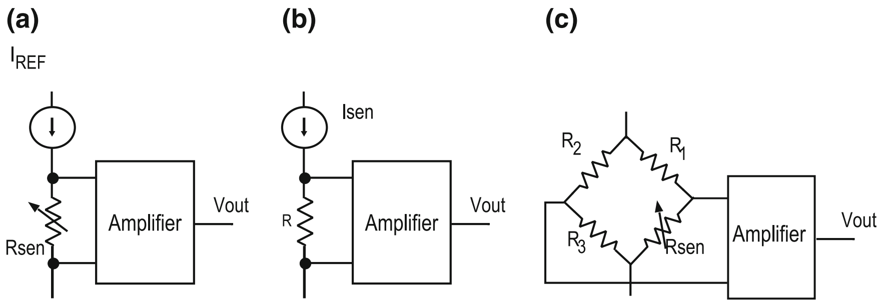


Fig. 10.10 Sensor conditioning examples with differential or instrumentation amplifiers: **a** for resistive type sensor, **b** for current type sensors; **c** with resistive sensors in bridge

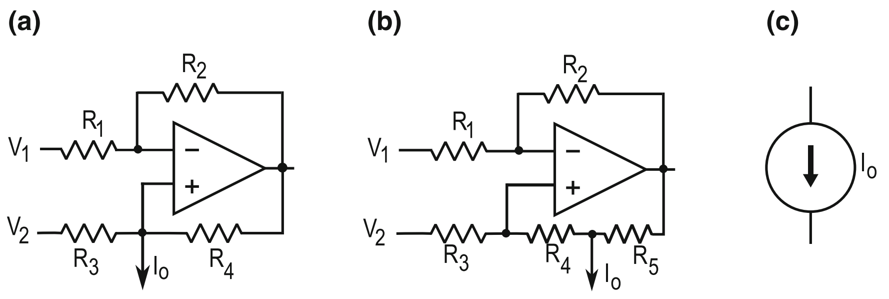


Fig. 10.11 Howland current source: **a** basic configuration, **b** modified improved version, **c** equivalent representation

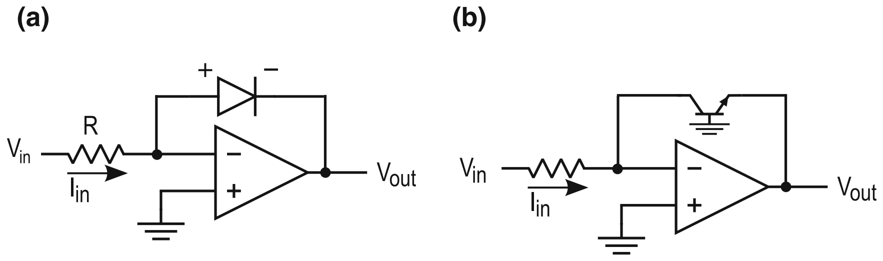


Fig. 10.12 Basic logarithm amplifiers: **a** diode based, **b** NPN bipolar based

find in literature several modifications to take into account limitations. Any input can be grounded, so positive or negative values for the current I_o are possible.

The OA also has non linear applications. One of them is using the device as a comparator, either by not using feedback, in which case saturation is reached easily because of the high OA gain, or by forcing a positive feedback loop into the OA.[2] This use is sometimes practical although it is better to utilize comparator circuits, which are based on the same OA transistor construction but are designed to always be in saturation.

Other non linear applications use the OA in the linear region, but include non linear elements in the circuit. One example is seen in the logarithmic amplifiers shown in Fig. 10.12. These amplifiers require the current I_{in} to be positive. Exchanging the resistance and the diode, or resistance and transistor, one gets exponential amplifiers.

The output relation for the logarithmic amplifiers are as follows. For the diode circuit,

$$V_o = -V_T \ln\left(\frac{I_{in}}{I_S}\right) = -V_T \ln\left(\frac{V_{in}}{RI_S}\right) \tag{10.10}$$

[2] In most applications, positive feedback is attained by connecting the OA output to the non-inverting input. However, the concept should be considered in the overall circuit, and some applications may use the local positive feedback connection while still providing a real negative feedback.

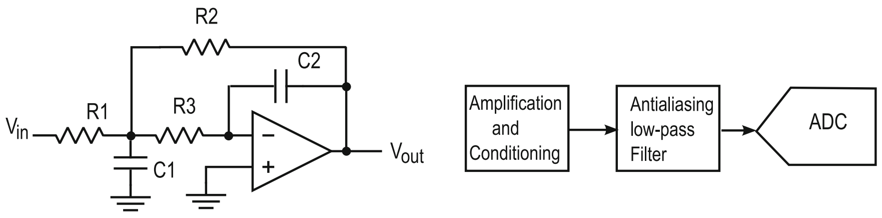


Fig. 10.13 A low-pass filter example and insertion for antialiasing

where I_S is the saturation diode current for (a), and the emitter-base junction saturation current in (b). V_T is the thermal voltage. The first circuit serves also as basis for a commercial IC logarithm amplifier from Texas Instruments, the Log114,[3] which has many applications, including interfacing with photo diodes and other sensors to ADC's.

10.4.2 Antialiasing Filter

OA based active filters, more specifically, antialiasing filters, are very important in the analog signal chain. Figure 10.13 illustrates an example of a low-pass filter and its insertion in the analog chain. Bandpass filters are also sometimes used. The importance will become clearer in Sect. 10.7.

The subject on OA and their applications in the analog signal chain is quite extensive. For a more in depth treatment of operational amplifiers in different applications and how to deal with imperfections, the reader may consult the references at the end of the book.

10.4.3 Operational Amplifiers in MSP430

Several members of the MSP430'x2 and 'x4 series have two or three internal OA as peripherals, denoted as OA0, OA1 and OA2. All OA's are single supply, rail-to-rail output. Through software the user can control their speed (slew rate), for optimized settling time versus power consumption, configure single or multiple OA circuits, including differential amplifiers, and also program PGA (Programmable Gain Amplifier) with a feedback resistor ladder. In addition, the MSP430'x4xx series allow to configure for rail-to-rail input.

Operational configurations are achieved by software via readable/writable control registers 0 and 1, OAxCTL0 and OAxCTL0. There are slight variations in the control

[3] See document SBOS301A at http://www.ti.com/lit/ds/symlink/log114.pdf for details and application examples.

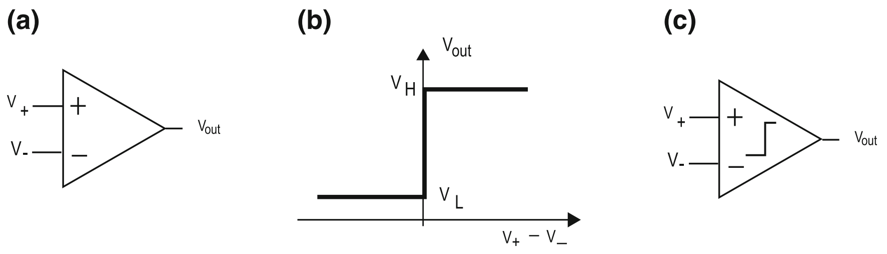


Fig. 10.14 Comparator: **a** symbol, **b** transfer curve, **c** alternate symbol

registers of the two series, as well as among different members of the 'x4xx series. Moreover, some members of this series also have switch control registers for the OA's.

In this text we explain briefly the control registers for the 'x2xx series, applicable to some members of the 'x4xx series as well. We recommend the reader to consult the user guides and data sheets for more information.

10.5 Comparators

A comparator is basically a high gain differential amplifier, like the OA. In fact, the OA can be used as comparator, but it is preferable to utilize ad-hoc circuits if possible. The symbol(s) and output curves are shown in Fig. 10.14. For many engineers, this is the most basic element in the signal chain.

Ideally, the comparator output voltage goes to a high value V_H whenever its non-inverting input is greater than the inverting, and to a low value V_L when it is lower. Input equality is undefined, as explained later. In mathematical terms,

$$V_{out} = \begin{cases} Vs_H & \text{for } (V_+ - V_-) > 0 \\ Vs_L & \text{for } (V_+ - V_-) < 0 \end{cases} \tag{10.11}$$

The values of V_{sH} and V_{sL} may be equal to the rail values or to other values by design. They are specified by the manufacturer.

Most comparators have either an open collector or open drain output, or else a push-pull output configuration. The former cases allows for connection to a subsystem with a different voltage supply. In this case, the output cannot be left floating; usually a pullup resistor is used.

A common application of comparators is that of threshold detector. Two simple examples are shown in Fig. 10.15. Circuit (a) is a simple threshold detector, which goes to VH whenever the input signal exceeds the threshold reference VREF. This operation is illustrated by Fig. 10.16. The circuit in Fig. 10.15b is a window comparator. Here, both comparator outputs are high only when $V_{\text{REFL}} < V_{in} V_{\text{REFH}}$.

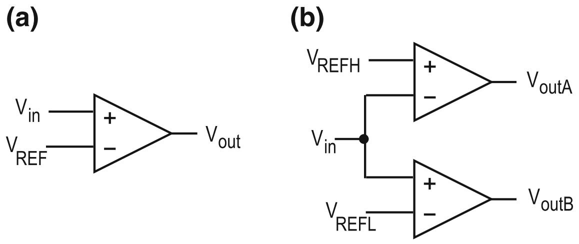


Fig. 10.15 **a** A basic threshold detector; **b** window comparator

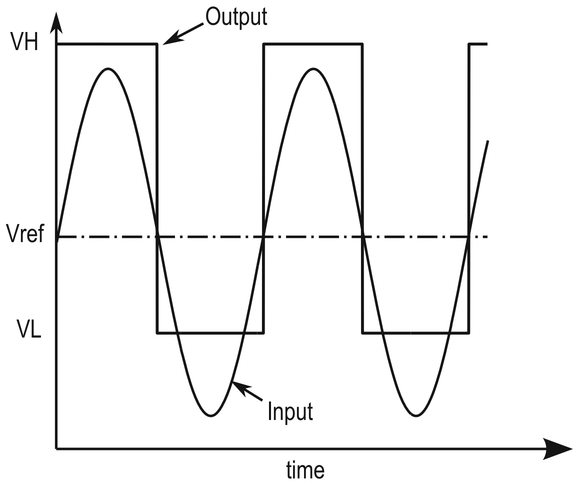


Fig. 10.16 Threshold detection with comparator

Comparator A goes low if the input signal exceeds the upper bound of this window, and comparator B goes low when it goes below the lower bound.

When the comparator operates with relatively low absolute differences between the two inputs, there are stability problems and the output oscillates. This also happens when processing slowly varying signals with even small amounts of superimposed noise. Noisy signals can occur in any application, and especially in industrial environments. To improve operation under these circumstances, *hysteresis* can be utilized. Also, it is possible to filter the output so that oscillation is limited to a narrower range of inputs.

In hysteresis comparator circuits, the transition between states occurs at different thresholds values V_{IH} and V_{IL} as illustrated in transfer function of Fig. 10.17. A comparator with this characteristic is said to be a *hysteresis comparator*, and it can be either non-inverting or inverting.

If the output in the non-inverting comparator is low, VL, it will stay there until $V_{in+} - V_{in-}$ increases and reaches the high threshold value VTH, when the output will change state to the high value VH. At this moment, the threshold for changing again to the low state VL is now VTL $<$ VTH. The performance for the inverting hysteresis comparator is explained in similar terms. The *hysteresis value* is defined as

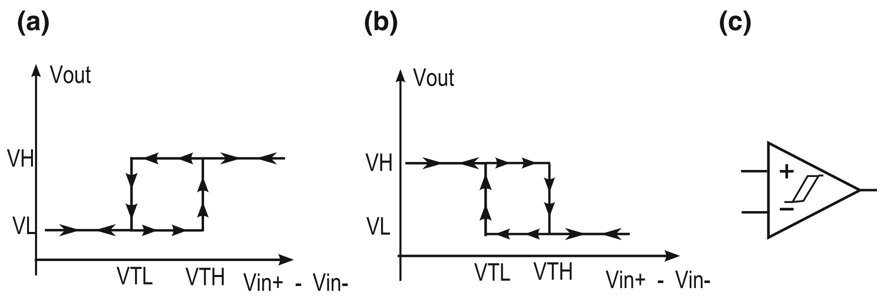


Fig. 10.17 Output for hysteresis comparator: **a** non inverting, **b** inverting, **c** symbol for a hysteresis comparator

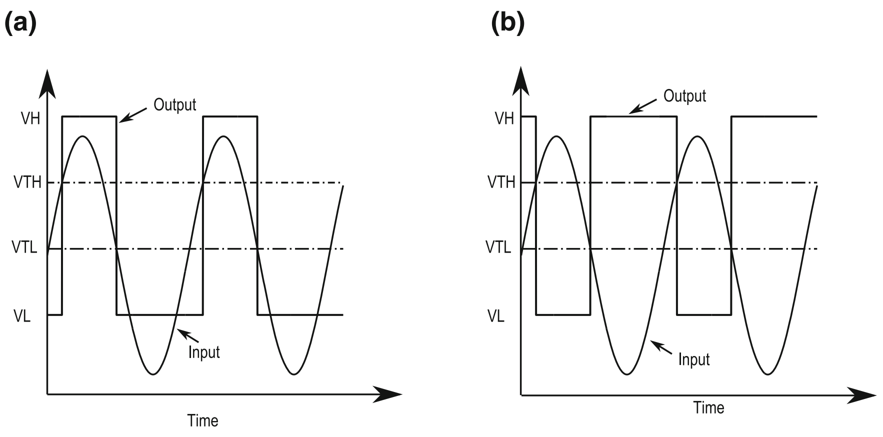
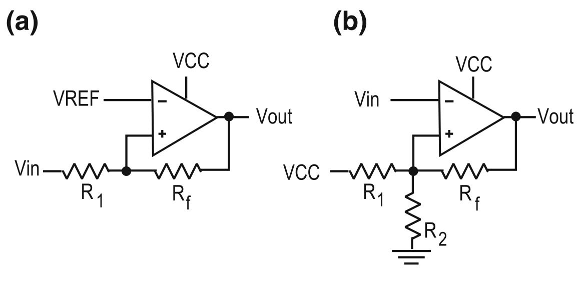


Fig. 10.18 Threshold detection with hysteresis: **a** non-inverting case, **b** inverting case

Fig. 10.19 Examples of hysteresis comparators: **a** non-inverting, **b** inverting



VTH-VTL. Figure 10.18 shows the changes in threshold detection when hysteresis is included.

Two comparator circuits with hysteresis are shown in Fig. 10.19. These positive feedback configurations are also applicable to OA with similar results. The non inverting circuit in this figure has the drawback of presenting a finite input impedance to the signal voltage, requiring an OA buffer if a high-impedance were required.

Let us now work some calculations, first for the non-inverting hysteresis comparator. From the nodal equations we find

$$V_{in+} = \frac{R_f V_{in} + R_1 V_{out}}{R_1 + R_f} \tag{10.12}$$

If the input voltage is large enough, the output will be V_H, and this state will remain while $V_{in+} > V_{\mathrm{REF}}$. Using (10.12), we arrive at

$$VTL = \left(1 + \frac{R_1}{R_f}\right) V_{REF} - \frac{R_1}{R_f} VH \tag{10.13}$$

Similarly, when the input is low, the output is VL and will remain while $V_{in+} < V_{REF}$. Hence,

$$VTH = \left(1 + \frac{R_1}{R_f}\right) V_{REF} - \frac{R_1}{R_f} VL \tag{10.14}$$

Therefore, the hysteresis value for the non inverting hysteresis comparator becomes

$$Hyst_{\text{non inverting}} = VTH - VTL = \frac{R_1}{R_f}(VH - VL) \tag{10.15}$$

For the inverting case, we work similarly starting with the equation

$$V_{in+} = \frac{R_f R_2 VCC + R_1 R_2 V_{out}}{R_1 R_2 + R_1 R_f + R_2 R_f} \tag{10.16}$$

Notice however that now the high and low threshold values correspond to outputs as VH and VL, respectively. Therefore,

$$VTH = \frac{R_f R_2 VCC + R_1 R_2 VH}{R_1 R_2 + R_1 R_f + R_2 R_f} \tag{10.17}$$

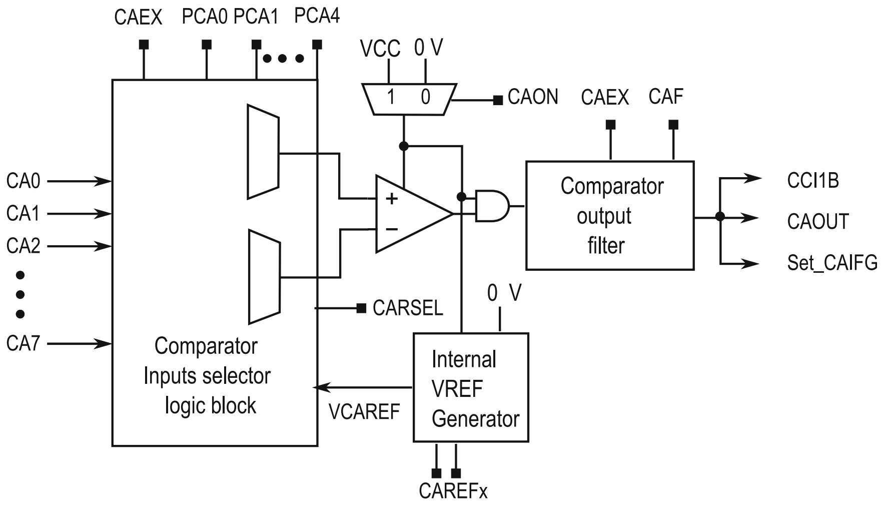
$$VTL = \frac{R_f R_2 VCC + R_1 R_2 VL}{R_1 R_2 + R_1 R_f + R_2 R_f} \tag{10.18}$$

$$Hyst_{\text{inverting}} = \frac{R_1 R_2}{R_1 R_2 + R_1 R_f + R_2 R_f}(VH - VL) \tag{10.19}$$

Example 10.4 *If VH = VCC and VL = 0 V, determine VTH, VTL and the Hysteresis value for the inverting and non inverting hysteresis comparators if all resistors are equal.*

Solution: *Substituting values in the above equations we have: (a) For the non inverting case:*

$$VTL = 2V_{REF} - VCC \quad VTH = 2V_{REF} \quad \text{and } Hyst = VCC$$



(Note: Comparator A provides only CA0 and CA1 -- PCA0 and PCA1 --)

Fig. 10.20 Block description of the comparators A and A+ in MSP430

(b) For the inverting case

$$VTL = VCC/3 \quad VTH = 2\,VCC/3 \quad and\ Hyst = VCC/3$$

Comparators are very important to the point that almost any ADC involves at least one device in its structure. Digital and mixed circuits usually contain comparators as well. Applications and considerations when dealing with comparators are discussed in literature.[4]

10.5.1 Internal Comparators in MSP430 (A and A+)

Several MSP430 models provide internal configurable comparators with interrupt capability. There are three models available: Comparators A, available in models of the MSP430x1xx family and some small models, Comparator A+ and, in recent families, Comparator B. We discuss briefly the first two models. The functional diagram for these comparators are shown in Fig. 10.20.

There are three registers associated to the MSP430 comparator as shown in Fig. 10.21. Control registers CACTL1 and CACTL2 allow to configurate the operation. Port Disable Register CAPD bits are used to disable the buffer associated to

[4] The reader may consult http://www.analog.com/library/analogDialogue/archives/34-07/comparators/comparators.pdf.

Fig. 10.21 Comparators A and A+ control registers CACTL1, CACTL2 and CAPD (Comparator Port Disable)

CACTL0

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit 0
CAEX	CARSEL	CAREFx		CAON	CAIES	CAIE	CAIFG

CACTL1

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit 0
CASHORT	P2CA4	P2CA3	P2CA2	P2CA1	P2CA0	CAF	CAOUT

Comparator A+ only ←

CAPD

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit 0
CAPD7	CAPD6	CAPD5	CAPD4	CAPD3	CAPD2	CAPD1	CAPD0

the external pin input used by the comparator. This is convenient when the inputs are analog, since it reduces power consumption. The registers are cleared at reset.

The registers are readable and writable, with the exception of bit0 in CACTL2. This bit, CAOUT, reflects the comparator output and is shared with an I/O pin. The output also feeds the interrupt subsystem as SET_CAIFG signal to trigger an interrupt request, as well as signal **CCI1B** for capture input of Timer A.

The configurable features of comparator A and A+ are the following:

1. The system can be turned on or off, with the CAON bit.
   - When off, there is no power consumption from this subsystem.
   - Inputs should not float when working.
2. Comparator inputs may be connected to external inputs CA0, CA1, …, or CA7, or to an internal reference voltage VCAREF.
   - Controllable by PCAx bits to connect to output I/O pin CAx.
   - CARSEL controls which the input to which VCAREF is applied.
3. The internal reference voltage VCAREF, when selected, can be set to 0.5VCC, 0.25VCC or diode voltage. The combination of bits CAREFx controls this feature. (CAREF0 puts VCAREF disconnected)
4. When working with analog inputs, the digital I/O pin x input buffer may be disabled with CAPDx.
5. The comparator output can be filtered with CAF to reduce oscillations.
6. In comparator A+, both inputs can be shorted with CASHORT and the comparator used for a sample and hold operation.

More detailed information about the MSP430 comparators can be found in the user guides and data sheets. The Comp_B module, not considered here, supports precision slope AD conversions, supply voltage supervision, and monitoring of external analog signals.

Example 10.5 *Consider using the comparator A+ in an MSP430g2553 for a simple threshold detection (see Fig. 10.15) with the analog signal connected to the +input and the reference voltage to the −input, using 0.5VCC as reference voltage. This value is available internally, so we can connect* **VCAREF** *to the negative input of comparator.*

Solution: *From the data sheet, we see that* CA1 *is shared with the I/O pin P1.1; the comparator is automatically used when the comparator input is selected. From the user guide we find that to connect* CA1 *to the +input we need* PCA4 = 1, PCA0 = 0, CAEX = 0*. To connect the internal voltage reference to the −input, CAEX = 0, CARSEL = 1. The 0.5VCC reference is set with* CAREF2 (=10)*. We disable the buffer for reduced power consumption. Thus, we configure with*

```
SET_CA  mov.b #CAON+CARSEL+CAREF2,&CCTL1 ; system ON, Vref=0.5VCC to - input
        mov.b #P2CA4,&CCTL2        ; P1.1/CA1 to + input
        mov.b #CAPD1,&CAPD         ; Input buffer disabled
```

A C-language version of the same code fragment:

```
CCTL1 = CAON+CARSEL+CAREF2  // system ON, Vref=0.5VCC to - input
CCTL2 = P2CA4               // P1.1/CA1 to + input
CAPD =  CAPD1               //  Input buffer disabled
```

10.6 Analog-to-Digital and Digital-to-Analog: An Overview

The digitalization of a continuous analog signal consists in assigning digital codes to discrete samples of an application segment. The concept follows the principles established in Chap. 2, Sect. 2.10.

A properly continuous conditioned signal within limits determined by the available hardware, is first sampled at specific time instants. Each sampled value falls within a certain subinterval to which an n-bit word is associated. This sequence of words represents the function in the given interval. A faithful representation depends on various conditions that include the sampling rate and the number of bits n, used to encode the discrete signal values as well as other parameters, we must comply with.

The quantized values are transferred to a digital system, whatever this is, where they are processed by the program or hardware system designed by the user. In many applications, the objective of the process could be to simply reproduce the function of interest in another target.

Example 10.6 *Consider Fig. 10.22a illustrating the output of the sensing stage in the millivolts range, intended to be used for an AD converter requiring a 0–5 V input range. Hence, the signal is amplified 98.33 times to obtain an appropriate reproduction in this range, obtaining the function shown in (b).*

Let us now encode this function with 4-bits. First, we sample the input signal at regular intervals. For our example, let us take 21 samples. The result is illustrated with black dots in Fig. 10.23. The x-axis values are now the integers corresponding to the order of the sample in the sequence of interest.

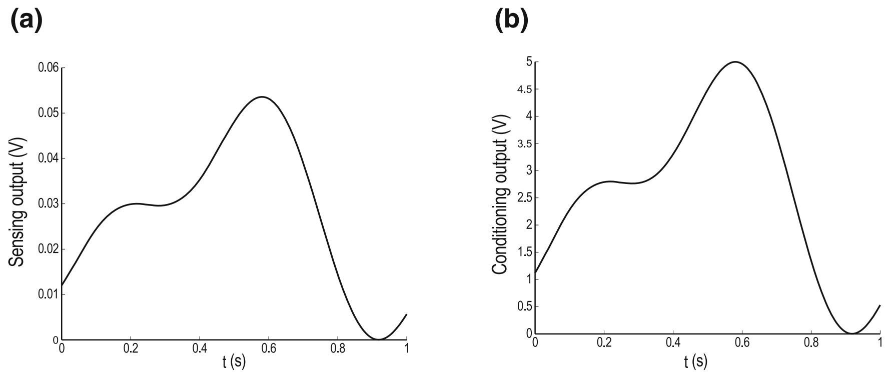
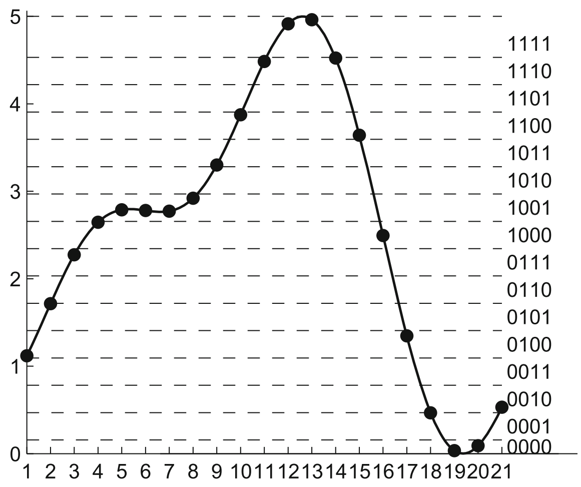


Fig. 10.22 From analog to the digital process: **a** a sensing output, **b** after conditioning

Fig. 10.23 Sampling and encoding with four bits the signal in Fig. 10.22b



With $n = 4$ *bits, we have* $2^4 = 16$, 4-*bit words. The voltage range (y-axis) is therefore subdivided in sixteen uniform subintervals, each of which is associated to a* 4-*bit code, as shown in the figure. Any sample falling within a subinterval is encoded with that code. For reasons to be explained later, the lowest and highest subintervals are not of the same size as the other ones.*

To each sampled value we associate now the corresponding code. For example, sample 1 falls in the subinterval tagged with 0100. *In this way, the function is then represented in digital form by the sequence of words* {0100, 0101, 0111, 1000, 1001, 1001, 1001, 1001, 1011, 1100, 1110, 1111, 1111, 1110, 1100, 1000, 0100, 0001, 0000, 0000, 0010}.

The process illustrated by the above example is called *quantization*. The inverse process is carried out by the digital to analog phase. The Digital-to-Analog converter takes an n-bit word and delivers an output value KV_{ref}, where K is proportional to the decimal equivalent of the digital word. This is done for each word in a sequence. An analog continuous signal will be constructed from the set. The accuracy will depend on the number of bits used in sampling, the number of samples, and the building method. The result may need to be further conditioned with filters, amplification/attenuation, or other operations. It may also be converted into a non-electrical magnitude using a transducer.

Example 10.7 *Let us reconstruct the function from the previous 4-bit sequence, which in decimal terms is* { *4,5,7,8, ..., 0, 2*}*, using a 5 V reference voltage and a proportional constant 1/2^n, where n is the number of bits. For example, the first point in reconstruction is* $4 \times 5/16 = 1.25\,V$*.*

*Figure 10.24 shows the reconstructed (circles) and sampled points (*x*). Although the figure looks like the function, there are differences between the two sets. The difference between the true value and the reconstructed one is called quantization error. As an example, the true value at* $t = 0$ *is* 1.1196 *while the value obtained is* 1.25.

This figure is discrete and we should fill the space between consecutive points with the intermediate values, that is, interpolate. Unprocessed results usually keep a constant value between reconstructions steps, yielding a curve like the one in Fig. 10.25a. One "smoothing" filter could provide straight line between consecutive points, as shown in (b), obtaining a piece-wise linear curve. There are of course other processing steps possible.

Although imperfect in this example, the process of going from a stepwise function to a continuous curve is called smoothing. As one might expect, a better result is obtained with more bits.

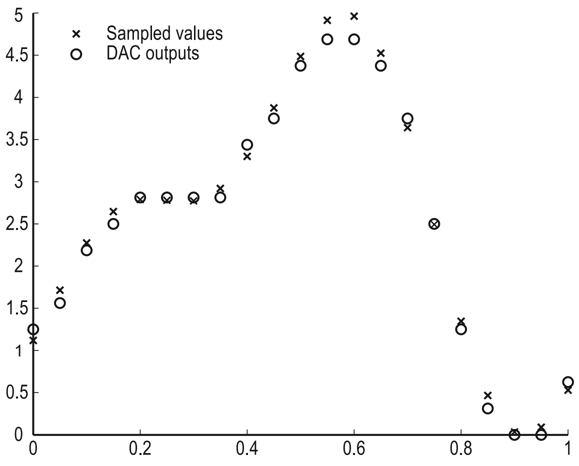


Fig. 10.24 Output values from DAC

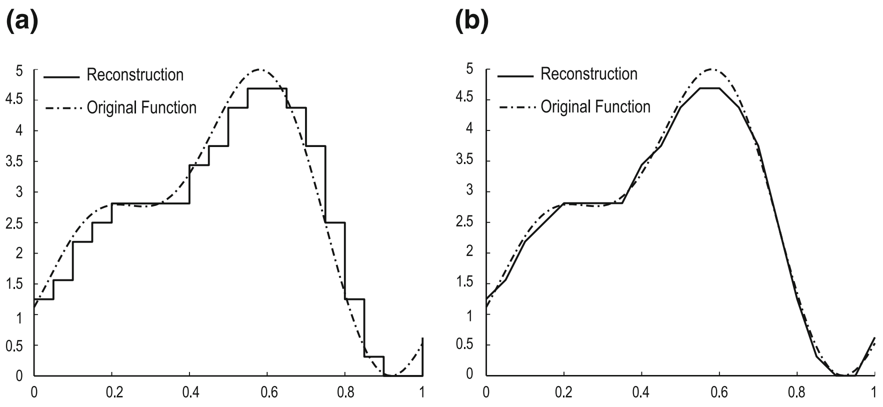


Fig. 10.25 Reconstructing the analog function of Fig. 10.22b using 21 samples and 4-bit words: **a** step-wise form; **b** piece-wise linear form

10.7 Sampling and Quantization Principles

As illustrated with the previous examples, *sampling* starts the conversion of an analog signal into a digital representation. The assignment of digital codes is called *quantization*, in which the analog function is represented by a set of discrete values, each identified by an n-bit word. Let us take a closer look at the sampling and quantization procedures.

10.7.1 Sampling and the Nyquist Principle

The time interval between two consecutive samples is the *sampling period* or *sampling rate* T_s. Its inverse is the *sampling frequency* $f_s = 1/T_s$, measured in Hertz or cycles per second. It is also common to use "*samples per second* (sps)" as units for this frequency. In Example 10.7, $T_s = 0.05$ s and $f_s = 1/.05\,\text{Hz} = 20\,\text{Hz} = 20\,\text{sps}$.

The first problem we should address is to take enough samples so as to be able to reconstruct back the function with enough accuracy. The answer to this problem was formulated by Henry Nyquist, from IBM Bell Laboratory, in the 1940s:

Nyquist principle: If f_h is the highest frequency component of a signal in its bandwidth of interest, then the sampling frequency f_s must satisfy

$$f_s \geq 2 \times f_h \tag{10.20}$$

The frequency $f_N = 2 \times f_h$ is called the *Nyquist frequency* and its inverse is the *Nyquist rate*.

The term "Nyquist rate" is in fact used by engineers both for the frequency and the associated period, the meaning being interpreted by context. Let us illustrate Nyquist principle with an example.

Example 10.8 *Consider a single sine wave function with frequency $f_h = 3$ kHz to be sampled. The Nyquist frequency is $f_N = 6$ kHz, with a Nyquist rate of $T_N = 167\,\mu$s. Nyquist principle states that for the function to be recovered adequately, the sampling rate must be less than or equal to T_N.*

We can see in Fig. 10.26a that for a rate of 1.3 times the Nyquist rate, the original function is not recovered at all. For the other two cases the reconstruction is better.

10.7.2 Sampling and Aliasing

Aliasing is the term used to describe the fact that when a signal of frequency f is sampled with a frequency f_s, the same samples will apply to any other signal of frequency $k\,f_s \pm f$, where k is an integer.

Figure 10.27 illustrates the aliasing phenomenon for a pure sine wave of frequency $f = 250$ Hz and a sampling frequency $f_s = 800$ Hz, with $k = 1$.

The problem of aliasing arises whenever the bandwidth of interest also contains components which we do not want to sample. The maximum frequency that might be sampled should comply with $f \leq \frac{1}{2} f_s$. For example, though audio frequencies are in the range 0–20 kHz, digital audio broadcast systems (DAB) take samples at 32 kHz, and thus they can only reproduce up to 16 kHz faithfully.

To avoid the inconvenience of reproducing unwanted signals, we use low pass and bandpass *antialiasing filters*.

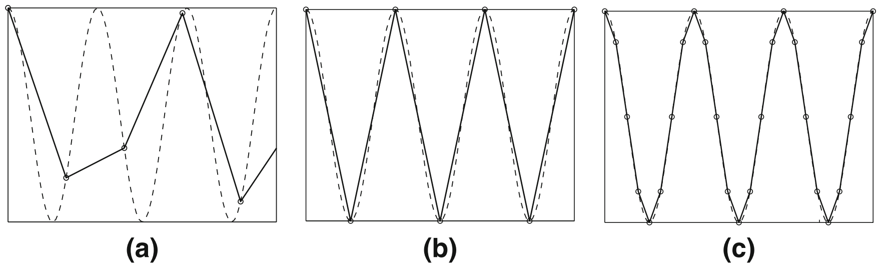


Fig. 10.26 Reconstructing from samples: **a** at 1.3 times T$_N$ ($fs = fN/1.3$); **b** at Nyquist rate, and **c** four times T$_N$ ($f_s = 4f_N$)

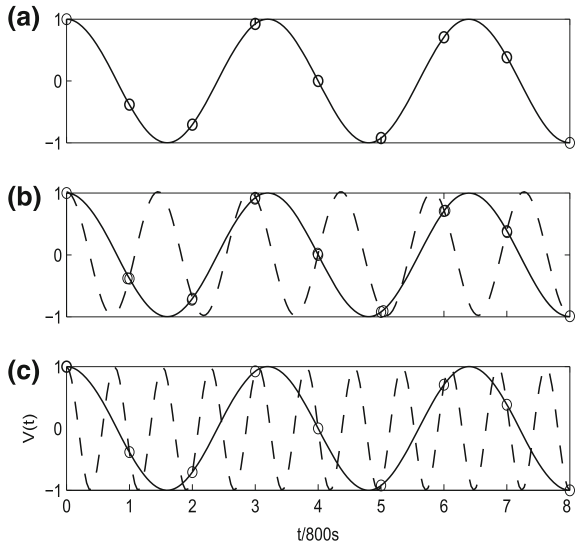


Fig. 10.27 Illustrating aliasing: **a** signal of frequency f being sampled at frequency f_s; **b** original signal and another with frequency $f_s - f$; **c** original signal and another with frequency $f_s + f$

10.7.3 Quantization

In addition to sampling frequency, the user must decide the size of the words to encode the data. With quantization, we represent a continuous interval by a set of 2^n discrete values, called *quantization levels* each one associated to an n-bit word. These levels are representatives for a set of 2^n subintervals, so that any analog value within a subinterval is encoded by the same word as the quantization level. This introduces an error in the process, called *quantization error* defined by

$$\text{Quantization error} = \text{actual value} - \text{quantization level} \tag{10.21}$$

The fundamental hardware devices that are used in the quantization process are the *Analog-to-Digital Converter* (ADC) and the *Digital-to-Analog Converter* (DAC). The first one associates the n-bit words to the analog value, the second one produces the analog value of the quantization level associated to a given n-bit word. The symbols for the ADC and DAC are shown in Fig. 10.28. Notice that the difference in the symbols is basically the direction of the information flow.

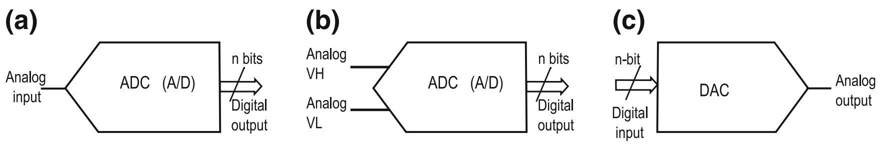


Fig. 10.28 Basic quantization: **a** ADC symbol; **b** differential input ADC; **c** DAC symbol

Data converters work with voltage (or current) inputs in a range $Vmin < Vin < Vmax$. This range is not always equal to the one delivered by the sensing stage, so pre-conditioning may be necessary. It is the responsibility of the designer to insure a proper relationship. For the hardware devices, it is common to have Vmin = 0 or Vmin = −Vmax. The *full scale range* V_{FS}, or simply FS, is defined by

$$VFS = Vmax - Vmin \tag{10.22}$$

Usually, the quantization levels are equally distributed in the range of interest. In this case, the difference between the levels is given by

$$\Delta = \frac{Vmax - Vmin}{2^n} = \frac{FS}{2^n} = \text{LSB} \tag{10.23}$$

Δ is called *resolution* or *precision* of the quantization. It is also called the *Least Significant Bit resolution*, or simply 1 LSB resolution. Notice that the resolution depends both on the full-scale FS and n. Thus, with FS = 5 V and $n = 8$, $\Delta =$ 19.5 mV, while FS = 3 V yields $\Delta = 11.72$ mV for the same n.

Because of the hardware characteristics, engineers usually prefer to talk of a resolution of n bits, instead of Δ. ADC's and DAC's have a fixed n value, but may accept different FS magnitudes.

From (10.23), the quantization levels are the 2^n values $Vmin + k\Delta, k = 0, 1, \ldots, 2^n - 1$. That is,

$$Vmin, \quad Vmin + \Delta, \quad Vmin + 2\,\Delta, \ldots, \quad Vmin + \left(2^n - 1\right)\,\Delta \tag{10.24}$$

Notice that $Vmax$ is not a quantization level, and no n-bit word is associated to it. Each quantization level is representative for a subinterval in $[Vmin, Vmax]$. The subdivision of this range can be done in different ways, two of which are illustrated in Fig. 10.29 for three bits, where subintervals are associated to 000, 001, …, 111.

Inset (a) is a quantization with all subintervals width equal to 1 LSB. Here, the first transition from subinterval 000 to 001 takes place 1 LSB after Vmin, and then each transition happens at 1 LSB intervals. Now, any value $x_j \leq x_{in} < x_{j+1}$ is encoded by the word for x_j, as illustrated in this figure for x_{in} to which we associate 010. In this example x_{in} is much closer to x_3. In fact, this subdivision has the disadvantage that the quantization error can be up to 1 LSB. To illustrate, consider an 8 V full scale, Vmin = 0 V, and a resolution of 4 bits, $\Delta = 0.5$ V. Then 0.49 V will be represented by 0000, i.e., 0 V.

Generally, a quantization error of 1 LSB = 1 Δ, is not acceptable. One solution is to modify the assignment of subintervals so that the maximum error is within $\pm\frac{1}{2}$ LSB. This is done by choosing the subintervals with the quantization levels as midpoints, as illustrated by Fig. 10.29b. In this case, the length for the first subinterval is $\frac{1}{2}$ LSB while the highest one is $1\frac{1}{2}$ LSB long. All the others are 1 LSB long. Therefore, with the exception of the last subinterval, the quantization error for

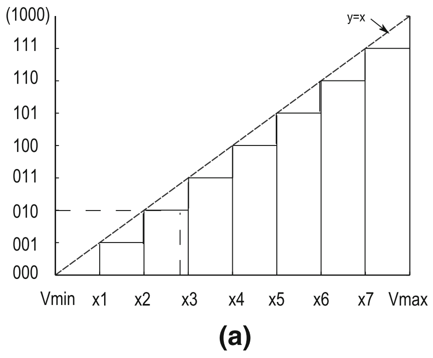


Fig. 10.29 Basic quantization: **a** all regular intervals; **b** half-LSB compensated (Levels: Vmin, x1, …x7)

any other value is within $\pm\frac{1}{2}$ *LSB*. This is the best solution, and thus this subdivision is the most common. Unless otherwise stated, this is the one we work with.

Figures 10.29a and b illustrate *straight binary codes*, where the lowest input voltage is assigned the all zeros word and then counting proceeds sequentially, using a full-scale input. Strictly speaking, this type of encoding is usually limited to unipolar voltages only, i.e., all voltages of the same sign. When $Vmin \neq 0$, it is also called *offset* or *biased binary code*.

In this situation, the decimal equivalent N_Q for the word assigned encoding given analog value x in the Half-LSB compensated subdivision would be

$$N_Q = \begin{cases} \left[2^n \frac{x - Vmin}{V_{FS}}\right] & \text{if } x < \text{Vmax and } [\bullet] < 2^n \\ 2^n - 1 & \text{if } [\bullet] \geq 2^n \end{cases} \tag{10.25}$$

where $[A]$ yields a rounding to the integer closest to the argument value A.

Example 10.9 *Assume that the interval* $[-10\,\text{V}, 10\,\text{V}]$ *is quantized with a resolution of* 4 *bits and regularly spaced quantization levels. (a) What is the LSB resolution; (b) List the quantization levels; (c) How do you encode* $-1.28\,\text{V}$ *in an all-regular subinterval quantization and what will the quantization error be? (c) Encode* -1.28 *with half-LSB compensated quantization and find the quantization error. (d) Find the LSB resolution with 12 bits, and the encoding and quantization error for* -1.28.

Solution: *Since* $n = 4$, *there would be* $2^4 = 16$ *subintervals.*

(a) The LSB resolution is found with (10.23)*:*

$$\Delta = \frac{10 - (-10)}{2^4}\,\text{V} = 1.25\,\text{V}$$

(b) The quantization levels are given as $Vmin + k\,\Delta$ *for* k $0, 1, \ldots, 2^n - 1$ *as:*

$$\begin{array}{llllllll} -10\,\text{V}, & -8.75\,\text{V}, & -7.50\,\text{V}, & -6.25\,\text{V}, & -5\,\text{V}, & -3.75\,\text{V}, & -2.50\,\text{V}, & -1.25\,\text{V}, \\ 0\,\text{V}, & 1.25\,\text{V}, & 2.50\,\text{V}, & 3.75\,\text{V}, & 5.0\,\text{V}, & 6.25\,\text{V}, & 7.50\,\text{V}, & 8.75\,\text{V} \end{array}$$

(c) Using (10.25), $[16 \times (-1.28 - (-10))/20] = [6.976] = 7$*, so the encoding yields* 0111. *The quantization value is* $-10 + 7 \times \Delta = -1.25$. *Therefore, the quantization error is* $-1.28\,\text{V} - (-1.25\,\text{V}) = -0.03\,\text{V}$.

(d) For 12 bits, the LSB resolution is $20/2^{12} = 4.8828\,\text{mV}$. *The LSB-compensated subdivision yields* $N_Q = 1{,}786$ *for an encoding* 0x6FA. *The associated quantization value is* -1.2793 *for a quantization error of* $-0.7\,\text{mV}$.

In straight binary codes, 1000…00 corresponds to the mid value. Very often, the minimum and maximum values are symmetrical around 0, with Vmin = −Vmax = −VR, as it was the case for the above examples. In these cases another encoding is generally preferred. Namely, a two's complement type, where the middle term 0000…00 is assigned to 0 V, and the distribution correspond to a two's complement representation in such a way that negative values are assigned a negative number, as illustrated by Fig. 10.30. This encoding is called *two's complement quantization* type, convenient for symmetrical bipolar ranges. The quantization levels in the x-axis have been labeled as shown to better illustrate the encoding scheme.[5]

It must be emphasized that the quantization levels used for an analog input is the same for both the offset and two's complement encoding cases, what is different is the encoding word. In twos complement quantization, the decimal signed integer N_Q value associated to an analog x is

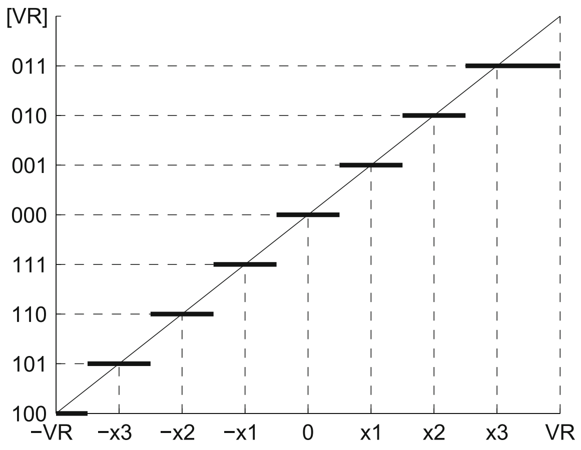


Fig. 10.30 Two's complement quantization

[5] When Vmin = −Vmax = $-V_R$, we have $\Delta = 2V_R/2^n = V_R/2^{n-1}$. Since the quantization levels are $-V_R + m\Delta$ for $m = 0, 1, \ldots, (2^n - 1)$, the value 0 will always appear as a quantization level.

$$N_Q = \begin{cases} \left[2^{n-1}\frac{x}{V_R}\right] & \text{if } x < +V_R \text{ and } [\bullet] < 2^{n-1} \\ 2^{n-1} - 1 & \text{if x} = V_R \text{ or } [\bullet] = 2^{n-1} \end{cases} \tag{10.26}$$

Conversely, given the decimal equivalent N_Q of an encoded word, the corresponding quantization level is

$$Level_{NQ} = N_Q\Delta \tag{10.27}$$

Example 10.10 *The interval in Example 10.9 is symmetrical and was encoded using an offset straight binary quantization. Now let us consider a two's complement code for the same range of values. (a) Verify that the quantization levels found using (10.27) are the same as before. (b) Encode* -1.28 V *for a* 4*-bit and a* 12*-bit resolution and find the quantization errors. How do they compare with the previous encoding?*

Solution:

(a) The values to consider for N_Q are $-8, -7, -6, \ldots, 0, 1, \ldots, 7$ since we are talking of signed numbers encoded with four bits. On the other hand, $\Delta = 1.25$*, as before. The reader can verify that the set of quantization values is the same by multiplying $N_Q\Delta$ for the different values of N_Q.*

(b) Using $n = 4$ in (10.26) we have

$$N_Q = \left[2^3 \times \frac{-1.28}{10}\right] = [-1.024] = -1$$

which gives a quantization level of $-1 \times 1.25\,V = -1.25\,V$, just as before. The corresponding code is 1111.

For twelve bits,

$$N_Q = \left[2^{11} \times \frac{-1.28}{10}\right] = -262$$

yielding a quantization level $-262 \times 4.8828\,mV = -1.2793\,V$, the same one as before. The corresponding code is 0xEFA.

Neither the quantization levels nor the quantization errors changed with the different encoding schemes.

Notice that the accuracy improves with more bits. However, there are physical limitations to the number of bits used, due both to hardware constraints and other factors such as the noise levels which can affect measurements.

10.7.4 Quantization and Noise

When the signal producing the x value to be quantized is static, the quantization error gives a good information of how well the translation process is realized. However, when the signal is varying with time, especially at high frequencies, the x value is more susceptible to be affected by noise. In this case, a better reference to consider *signal-to-noise ratio* (SNR). In general, SNR is defined in dB as

$$\text{SNR} = 20\log\left(\frac{\text{RMS value of signal}}{\text{RMS value of noise}}\right)\ \text{dB} \tag{10.28}$$

The higher SNR the better. Now, even in an ideal quantization process, there is a finite SNR that will depend on the LSB resolution and the magnitude of the signal.

For perfectly quantized signals, the reference is a sinusoidal of magnitude V_m, so the interval to quantize is $[-V_m, V_m]$. Using n bits in two's complement quantization, we infer that $V_m = 2^{n-1}(LSB)$ and the RMS value of the signal is then

$$\text{RMS of signal} = \frac{V_m}{\sqrt{2}} = \frac{2^{n-1}\text{LSB}}{\sqrt{2}} \tag{10.29}$$

On the other hand, for quantized signals it has been shown that the RMS noise is LSB/$\sqrt{12}$ = LSB$/(2\sqrt{3})$. Substituting this result and (10.29) into (10.28) we have

$$\text{SNR} = 20\log\left(\frac{2^{n-1}LSB/\sqrt{2}}{LSB/2\sqrt{3}}\right) = 20\log\sqrt{\frac{3}{2}} + n \times \log 2 \tag{10.30}$$

This translates into

$$\text{SNR} = 1.76 + 6.02n\ \text{dB} \tag{10.31}$$

This result applies to an ideal quantization process, independent of the hardware itself, and as such is the best value we can achieve. For this reason, this is a reference for design, since non idealities from hardware will degrade this figure. These factors are considered next.

10.7.5 Quantization and Hardware Imperfections

The above theoretical discussion related to sampling and quantization process not only introduces the theoretical concepts, but it applies to an ideal signal-chain composed by ideal hardware components. Let us now look into the errors resulting from the use of real devices. The usual errors, and figures of merit, to look at are summarized next:

Offset error: Any deviation from the point where the input voltage cause a transition from zero count. The offset error can usually be factored or calibrated out. Offset error may be expressed in percent of full scale voltage, Volts or in LSB. It is also called Zero Scale error.

- *Bottom offset* is the input required to cause the transition to the first count.

Full scale offset error: Is the error in the actual full-scale output transition point from the ideal value. It is also expressed in percent of full scale voltage, Volts or in LSB.

Gain error: It is the deviation from the ideal slope of the transfer function. Although the ideal slope is usually 1, it may perfectly be otherwise, a gain G.

Differential Non Linearity (DNL): This feature is also called Differential Linearity Error (DLE). This is the difference between the ideal and the actual *input code width*, the range of input values that produces the same digital output code. It is usually expressed in terms of the widest (positive) DNL and narrowest (negative) DNL. DNL is expressed in terms of LSB.

Integral Non Linearity (INL): This feature is also called Integral Linearity Error (ILE). This term describes the departure from an ideal linear transfer curve. INL does not include quantization errors, offset error, or gain error. Expressed in terms of LSB.

Signal-to-Noise Ratio (SNR): In addition to the maximum quantization SNR described by (10.31), the actual figure of merit includes the noise generated by hardware, the input circuit noise, and the jitter. These factors degrade the figure of the ideal case (10.31).

Total Harmonic Distortion (THD): This one measures the effect of harmonics due to non-linearities. If the input signal were assumed to have a fundamental component at frequency $f1$, quantization would produce harmonics at frequencies $f2$, $f3$, ... THD would then be measured using the RMS values as defined by (10.32) below. This definition excludes DC components.

Signal-to-Noise and Distortion (SINAD): This is a combination of SNR and THD, as defined in (10.33). It usually tracks better SNR, and as such is used to define the next figure of merit.

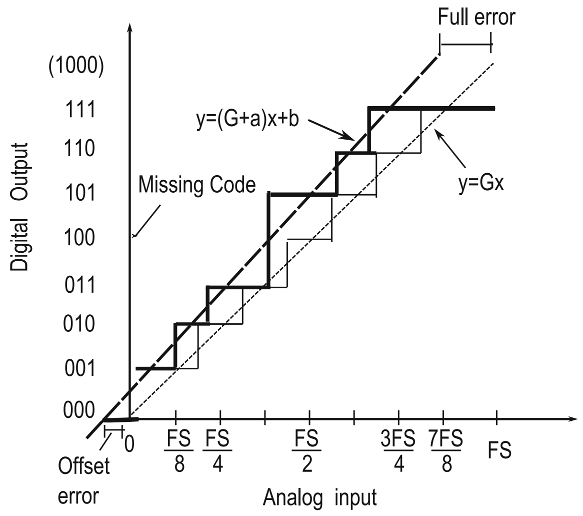


Fig. 10.31 Ideal versus actual quantization: gain error a, missing code 100

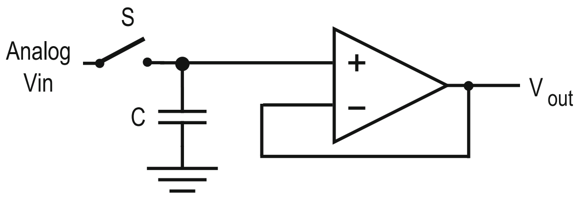


Fig. 10.32 Basic sample-and-hold circuit configuration

Effective Number of Bits (ENOB): Using the association between SINAD and SNR, this figure of merit states the number of bits of an ideal converter yielding a similar quantization. ENOB is defined by Eq. (10.34).

$$\mathrm{THD} = \sqrt{\frac{V_{f2}^2 + V_{f3}^2 + V_{f4}^2 + \ldots}{V_{f1}^2}} \tag{10.32}$$

$$\mathrm{SINAD} = -10\log\left[10^{-\mathrm{SNR}/10} + 10^{\mathrm{THD}/10}\right] \tag{10.33}$$

$$\mathrm{ENOB} = \frac{SINAD - 1.76}{6.02} \tag{10.34}$$

In the SINAD equation, notice that if the THD is sufficiently small, SINAD $\approx$ SNR. This is common with modern technology. If SINAD = SNR, then ENOB = n, the number of bits.

Other definitions are illustrated in Fig. 10.31.

10.7.6 Sample & Hold Circuits

A *sample-and-hold circuit* (SH circuit) samples the analog value to be processed by the ADC for conversion into an n-bit word. The basic principle is shown in Fig. 10.32. Some versions add a buffer before the switch to prevent loading effects. Ideally, when the switch S is closed at instant $t = t_k$, the input value $V_{in}(t_k)$ is stored at the capacitor C. This is the sampling phase. The switch S then opens to enter to the hold phase so the capacitor holds the stored value, which is readable at the output of the right buffer. The output buffer prevents the capacitor from discharging through the input resistance of the load. The switch S is a MOSFET switch, controlled by a clock with a sampling frequency f_s. Thus, samples are taken at $1/f_s$ intervals.

The above ideal description does not take into account the switch resistances, for example. There are many limitations that degrade the performance of the circuit. For example, the capacitor has leakage currents that discharge it, as well as actual currents going into the buffer, which are small, but not zero. Timing problems, imperfections in the switches and so on result in an unavoidable degradation of the signal, but still good enough for all practical purposes. Therefore high quality capacitors and fast switches are key elements in this operation.

One of the timing problems associated with sampling is *jittering*, which affects the periodicity of sampling and therefore introduces an instantaneous signal error. This error is proportional to the slew rate of the desired signal and the absolute value of the clock error, that is,

$$\Delta V = \left|\frac{\mathrm{d}V}{\mathrm{d}t}\right| \Delta t \tag{10.35}$$

For example, using N bits to sample a sine signal of frequency f, $(2^n\, LSB)$ sin $(2\pi f\, t)$ with an error less than 1 LSB, we must maintain the time deviations Δt such that it satisfies

$$\Delta V = \left|2^N\, LSB\, 2\, f\, \pi\, f \cos(2\pi f\, t)\right|_{t=0} \Delta t < 1\, LSB \tag{10.36}$$

The instant $t = 0$ applies because this is where the slope is maximum. From this expression, then

$$\Delta t < \frac{1}{2^{N+1}\pi\, f} \tag{10.37}$$

To illustrate, sampling with 10 bits an audio frequency in the upper range of, say $f = 20$ kHz, needs to keep an error of time sampling less than 7.8 ns to be adequate. Remember that the sampling frequency must be greater than or equal to 40 KHz, with a sampling period of 25 μs, and therefore the absolute deviation with respect to sampling instants must be kept within 0.03 % to insure fidelity. In CD's, sampling is done with 15 bits, requiring an error less than 0.25 ns.

As seen with the above figures, jittering is particularly critical in high-frequency signal conversion, or where the clock signal is prone to interference. For this reason, it is more common to work this problem with hardware instead of software. In the MSP430, for example, converters work together with internal timers. Delta-sigma ADC's suffer less with this problem because they are slower.

Practical commercial SH circuits (amplifiers) are available in IC form which are designed to minimize jittering. National Semiconductor's—now TI—LF398-n is a good example, where the holding capacitor is provided by the user. The Maxim's DS1843 consists of a fully differential sampling capacitor, switches, and a differential output buffer which can be configured for single-ended operations. For applications requiring multiple channels, Maxim's versatile MAX5167 offers 32 buffered sample/hold circuits with internal hold capacitors.

Nowadays, most ADC integrated circuits have the SH circuit already incorporated. This is also the case in MSP430 microcontrollers.

10.8 Digital-to-Analog Converters

As mentioned before, an n-bit *digital-to-analog converter* (DAC) receives as input n digital bit signals and delivers an analog value. This one is proportional to a decimal equivalent N_D of the binary word and a reference voltage V_{REF}, which is taken as the full-scale value V_{FS}. The input-output relation is given as:

$$V_{out} = K\,\frac{N_D}{2^n}V_{REF} \tag{10.38}$$

where N_D is the decimal equivalent to the input n-bit word. It is generally between 0 and $2^n - 1$, but signed two's complement equivalent is possible too. DACs working with two's complement coding inputs, are usually referred to as *bipolar DACs*. Notice the similarity with the definition of resolution in Eq. (10.23).

From (10.38) we can infer that a reference voltage signal is required. Modern DACs allow the user to select between an external reference or another one already built in the device. This reference is separate from the power supplies VDD and VSS needed. Terminals for these connections are often incorporated into the DAC symbol.

The DAC's output can be a voltage or a current signal. In the first case, the gain K depends on the hardware circuit used. For a current output it is possible to set this gain with Ohm's law using a resistor connected to the output port; current-to-voltage converters are another alternative. The resistor values are limited by the acceptable output voltage and the current capacity of the DAC.

The n bits at the input may be received all at once, as it is the case for parallel DACs, or else the bits are received one by one, for serial DACs. Both serial and parallel commercial DACs may be one or several channels type, meaning by this that inputs can be received from several sources and conversion outputs delivered at different terminals.

There are different types and architectures for DACs.[6] We mention one architecture which is very popular in DAC circuits, the R-2R ladder type.

10.8.1 R-2R Ladder

Figure 10.33a shows the basic configuration for an R-2R ladder. The figure shows a ladder for 4-bit words, but any number of bits can be considered by extending the R-2R ladder. A characteristic of this configuration is that the equivalent resistance seen to the left of any node n_j is 2R.

[6] Reference [1] provides a good overview of this topic.

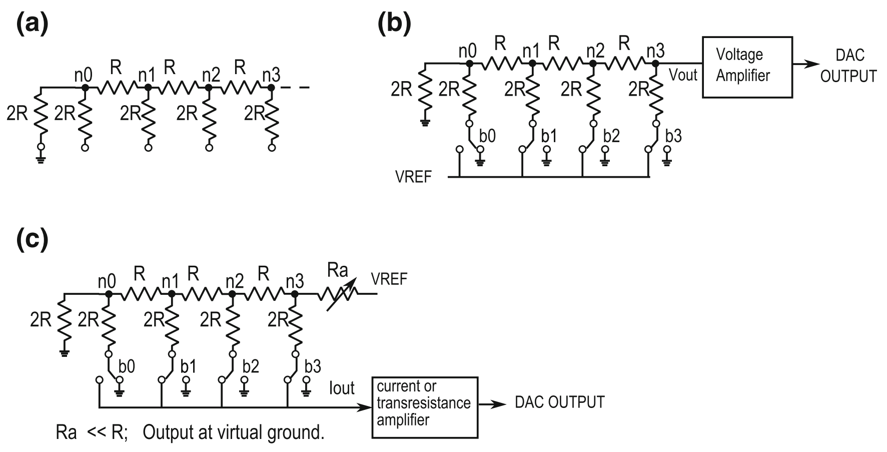


Fig. 10.33 R-2R ladder configuration

This ladder is so practical that commercial ICs with just the ladder (with some variations to add flexibility) are available from several vendors.[7]

Figure 10.33b shows how the circuit is used in voltage output DACs. The switches are operated by the input, with switch b_j connected to ground when bit $b_j = 0$ and to V_{REF} when the bit is 1. The case 1110 is illustrated in the figure. The Thevenin's equivalent for the circuit seen from the output gives a voltage

$$Vth = \frac{VREF}{2^n}\left(2^{n-1}b_{n-1} + 2^{n-2}b_{n-2} + \ldots + 2\,b_1 + b_0\right) \tag{10.39}$$

On the other hand, Fig. 10.33c shows how the circuit is used in current output DACs. Here the output is taken at a virtual ground node, which can be obtained through a current-to-voltage amplifier or a current amplifier. The input resistance seen by *VREF* is R + Ra. Assuming $Ra \ll R$, the input current will be $Iin \approx VREF/R$. The output current will then be given by an equation similar to (10.39).

More suitable for integrated CMOS technology, popular DAC configurations are based on capacitor arrays, the charge distribution DAC, like the one shown in Fig. 10.34.

In this configuration, switches b$_0$, b$_1$, … are controlled by the word bits, Vout is taken from a unity gain amplifier to prevent loading to the capacitors. The equivalent capacitance when all are in parallel is 2^nC. Conversion takes place in two phases. In the first one, all capacitors are discharged by closing S and connecting all switches to ground. In the second phase, S is open and the input digital word connects the switches to ground (for $b_j = 0$) or to VREF (for $b_j = 1$) to produce the analog

[7] For example, the QS009 and QS014 from TTE Electronics, BCN 31, and 68 Ladder from BI Technologies—later acquired by TT, or the SWR2R series from Semiconwell, ranging from 4 to 20 bits.

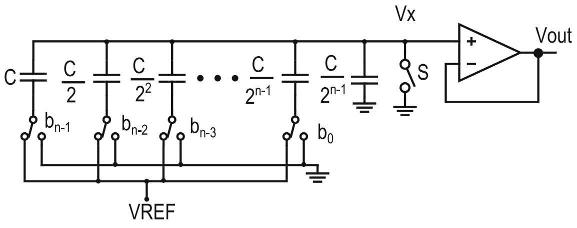


Fig. 10.34 Charge distribution DAC

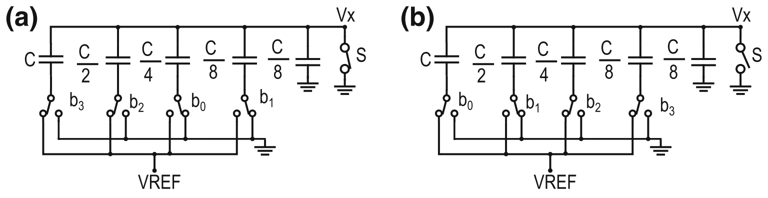


Fig. 10.35 Charge distribution DAC example. **a** Discharging phase; **b** Conversion phase

equivalent Vx. Observe that the sum of all capacitors is

$$C + \frac{1}{2}C + \frac{1}{2^2}C + \ldots + \frac{1}{2^{n-1}}C + \frac{1}{2^{n-1}}C = 2\,C \tag{10.40}$$

When all switches are connected to ground for the discharging phase, the total charge becomes $Q_T = 0\,$C. In the second phase, the conversion phase, the addition of the individual charges must not change, that is

$$\sum_{j=0}^{n-1} \left(b_{n-1-j}\,VREF - V_x\right)\frac{C}{2^j} + (0 - V_x)\,\frac{C}{2^{n-1}} = Q_T = 0$$

After some Algebra and using (10.40) we arrive at

$$V_{out} = V_x = \frac{VREF}{2^n}\left(b_{n-1}\,2^{n-1} + b_{n-2}\,2^{n-2} + \ldots + b_1\,2^1 + b_0\right) \tag{10.41}$$

This configuration has the disadvantage that the ratio between the largest and the smallest capacitor is $2^n - 1$, which is unacceptable and gives rise to many problems. Several good solutions to this problem have been found, but discussion of them is beyond the scope of this book.

Example 10.11 *Let us consider the charge distribution DAC with four bits, and convert* 1100*. The two phases are illustrated in Fig.* 10.35*a, b. The buffer is not included in the figure.*

In the second phase, we can write

$$(VREF - Vx)C + (VREF - Vx)\frac{C}{2} + (0 - Vx)\frac{C}{4} + (0 - Vx)\frac{C}{8} + (0 - Vx)\frac{C}{8} = 0$$

That is,

$$Vx = \frac{3}{4}VREF \Rightarrow Vx = \frac{12}{16}VREF$$

12 *is the unsigned decimal for* 1100 *and Vx has the form* (10.41).

10.8.2 Smoothing Filters

When the application is dynamic, the input words to the DAC will change periodically. This will result in a stepwise output like the one shown in Fig. 10.25a. To modify this appearance, the DACs output is fed into a *smoothing filter*, also called reconstruction filter, so that the output become like that of Fig. 10.25b.

10.8.3 Using MSP430 DACs

With the exception of the ’x3xx series, several members of the other series have one or two DACs incorporated. Series ’x1xx,’x2xx and ’x4xx have model DAC12, while ’x5xx and ’x6xx have model DAC12_A. The block structure of the DAC12 is shown in Fig. 10.36. The DACs are called DAC12_0 and DAC12_1, when two are present. The block structure is similar for both DACs, except that for DAC12_1 the control signal DAC12GRP is don’t care. Models with only one DAC do not include the DAC12_1 block nor the Group Load Logic Unit.

The DAC12 output is given as

$$\text{DAC\_Output} = \text{K} \times \text{VREF}\frac{\text{DAC12\_xDAT}}{2^n} \tag{10.42}$$

where the dynamic range K, the voltage reference VREF, and the resolution n can all be defined by the user. Although register DAC12_xDAT is 16 bits wide, only 12 or 8 are meaningful depending on the value of n. This register may be buffered through register DAC12_xLatch or directly applied to DAC and also receive data from the Analog to Digital converter ADC12.

Configuration is achieved through control registers DAC12_xCTLC shown in Fig. 10.37. Series ’x5/6xx DAC12_A has in addition a control register DAC12_xCTLC1, calibration control register DAC12_xCALCTL, a calibration data register DAC12_1CALDAT, and an interrupt vector generator register DAC12IV. These registers are all readable and writable, and are cleared on reset with the exception of DAC12_xCALCTL which is set to 9601h.

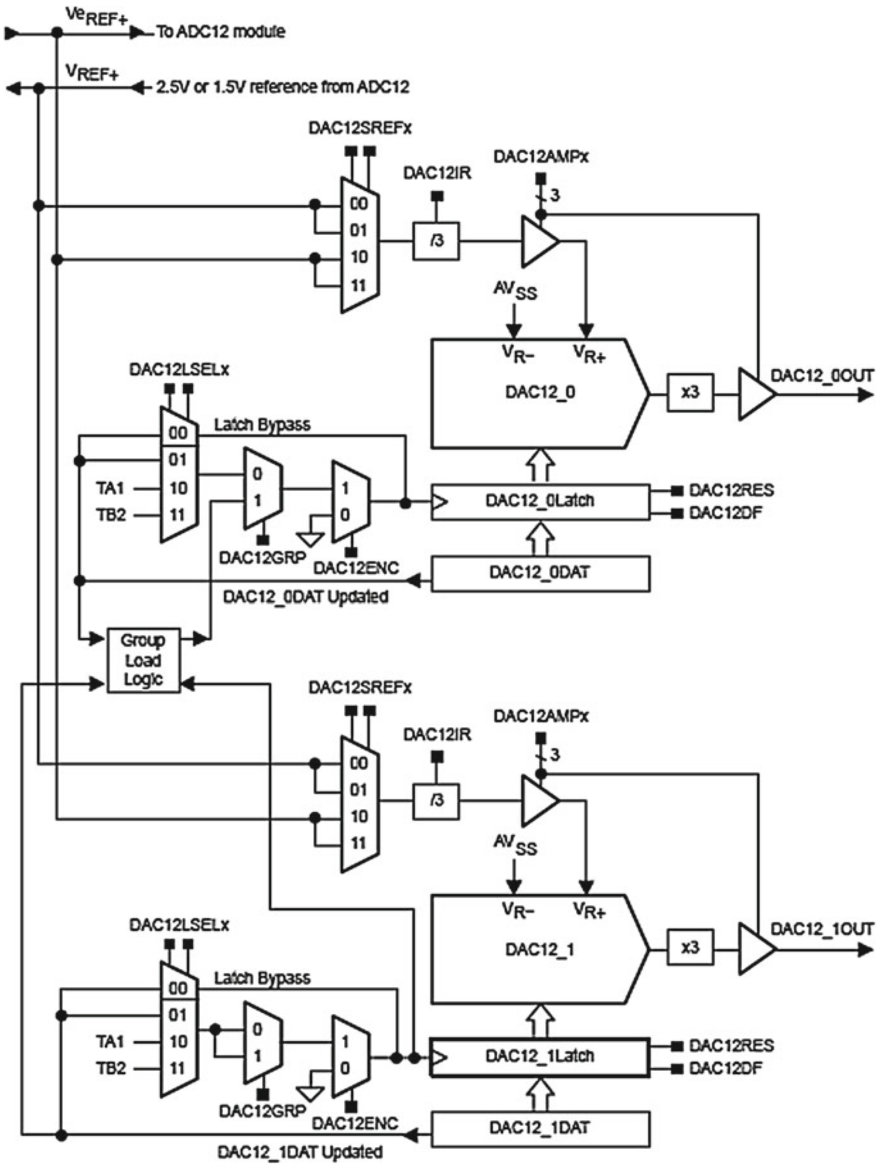
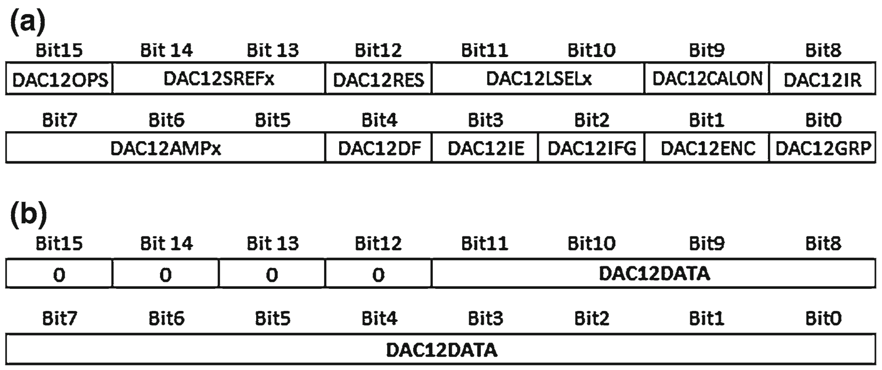


Fig. 10.36 MSP430 dual DAC configuration

MSP430 DACs have interrupt capability. Yet, the interrupt flags DAC12GIF shares the interrupt vector address with th DMA, so the flag is not automatically reset. The flag is set when a conversion is achieved. Table 10.3 shows features of the different families.



(a)

Bit15	Bit 14	Bit 13	Bit12	Bit11	Bit10	Bit9	Bit8
DAC12OPS	DAC12SREFx		DAC12RES	DAC12LSELx		DAC12CALON	DAC12IR

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DAC12AMPx			DAC12DF	DAC12IE	DAC12IFG	DAC12ENC	DAC12GRP

(b)

Bit15	Bit 14	Bit 13	Bit12	Bit11	Bit10	Bit9	Bit8
0	0	0	0	DAC12DATA			

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DAC12DATA							

Fig. 10.37 MSP430 DAC12 control register (DAC12_xCTL)—x = 0 or 1

Table 10.3 Digital to analog converter DAC12, DAC12_A in MSP430

Feature	’x1xx	’x2xx	’x4xx	’x5/6xx
8- or 12-bit resolution	Yes	Yes	Yes	Yes
Straight binary or 2’s complement data format	Yes	Yes	Yes	Yes
Data right/left justification	No	No	No	Yes
Adjustable dynamic range K	Yes	Yes	Yes	Yes
Values:	(1 or 3)	(1 or 3)	(1 or 3)	(1, 2 or 3)
Internal or external VREF selection	Yes	Yes	Yes	Yes
Internal Output availability*	No	Yes	Yes	Yes
Offset calibration capability	Yes	Yes	Yes	Yes
Data calibration capability	No	No	No	Yes
Interrupt vector generator	No	No	No	Yes
Programmable settling time versus power	Yes	Yes	Yes	Yes
Multiple DAC’s synchronized update	Yes	Yes	Yes	Yes

Example 10.12 *Let us use an MSP430FG4617 to generate* $\approx$1 V *from DAC12_0, using as reference voltage an external 3.3 V, with no extended dynamic range. From* (10.25)*, we find that the corresponding encoding* N_Q *is* 1241 = 4673 h.

From the use’r guide ([slau056j]) *we obtain the following information for the register bits:*

DAC12OPS = 0 *for output at* P6.6*, which comes by default at reset;*
DAC12SREFx bits = 10 or 11 *for the external reference;*
DAC12RE = 0 *for* 12*-bit resolution, which comes by default at reset;*
DAC12OPS = 0 *for output at* P6.6*, which comes by default at reset;*
DAC12IR = 1 *for one time reference voltage resolution.*

Finally, let us work with medium speed and current for both input and output buffers, DAC12AMPx bits = 101*. We are now ready to go:*

```
DAC120_setup    mov #DAC12REF_2+DAC12IR+DAC12AMP_5,&DAC12_0CTL
                                         ; Vref=3.3 external
```

```
DAC12_out       mov #1241,&DAC12_0DAT       ; 1 V at output
```

or in C

```
DAC12_0CTL = DAC12REF_2+DAC12IR+DAC12AMP_5
                          // DAC12_0 set up  Vref=3.3 external
DAC12_0DAT = 1241   // 1 V at output
```

The data register is bypassing the latch one, so the value of DAC12ENC *is a don't care. Many programmers prefer to set this bit.*

10.9 Analog-to-Digital Converters

An *Analog-to-digital Converter* (ADC, A/D), performs the opposite operation from the DAC, that is, it receives an analog input signal, and delivers an n-bit word. Most commercial ADCs range from 6-bit to 24-bit words, available from different vendors. Moreover, parallel and serial output ADCs are also in the market, with the latter ones being increasingly more popular.

There are different types of converters. Flash ADCs are the fastest ones, but very power hungry. The basic principle consists in identifying the subinterval of [0 V, VREF] in which the input value falls. To do that, it uses $2^n - 1$ comparators, whose outputs are input to a logic block that generates the n-bit word. An 8-bit ADC requires 255 comparators, for example. This structure has the great advantage of speed, but is limited to low number of bits, due to its size and power consumption. To deal with these disadvantages, several modifications and combinations have been proposed in the literature, such as the two-step ADC, pipeline ADCs, and others. Nevertheless, their discussion is out of the scope of this book.

There are several other architectures available. We discuss briefly three of them: The slope, the successive approximation, and the sigma-delta types. These architectures are also used as embedded peripherals in many microcontroller families.

10.9.1 Slope ADC

The simplest form of an ADC is the *slope ADC*, also called *integrating ADC*. Two versions are illustrated in Fig. 10.38, the single slope and the dual slope ADCs.

The logic units in these ADCs contain counters, output registers, the necessary logic to generate the signals to control the switches, synchronization, and so on. The Vin value is obtained from time measurements. In the single slope case, it measures the time t_u that it takes the integrators output to reach the Vin value. On the other hand, the controller of the dual slope ADC connects the integrators output to −Vin during a fixed time t_{uf} and then switches to VREF. We measure the time t_d that the integrator needs to get back to 0 V. These principles are illustrated in Fig. 10.39.

In the single slope ADC, the moment in which the integrator's output V_S reaches Vin is detected with a threshold detector. The comparator's change of state tells the digital unit to capture the word from a counter, and to generate a reset signal for the integrator. In this case, the time t_u is found from

$$V_{in} = \frac{\text{VREF}}{RC} t_u \tag{10.43}$$

This requires precision passive values and a good clock of stable frequency f_{clk}.

In the dual slope ADC, after t_{uf} seconds the integrator's output is

$$V_S = \frac{\text{Vin}}{RC} t_{uf} \tag{10.44}$$

The switch connects then to VREF and the measured time t_d for the integrator's output to become 0 V is related to the equation

$$-\frac{\text{VREF}}{RC} t_d + V_S = 0 \tag{10.45}$$

Using this information we have

$$\text{Vin} = \frac{t_d}{t_{uf}} \text{VREF} \tag{10.46}$$

The main disadvantage of slope ADCs is their speed. However, when speed is not a great concern, they have the advantage of simplicity, and can be realized with the comparators embedded in the microcontroller.

10.9.2 Successive Approximation ADC

A popular architecture, embedded as a peripheral in many microcontrollers, is the *successive approximation ADC*. The n bits are determined from MSB to LSB in n

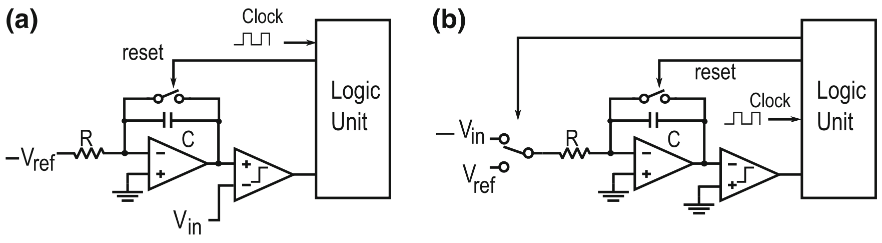


Fig. 10.38 Basic diagrams of (**a**) single slope ADC, and (**b**) dual slope ADC

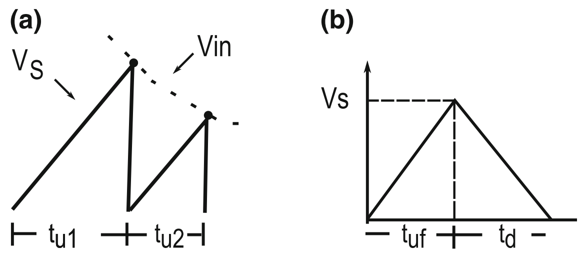


Fig. 10.39 Operation of **a** single slope ADC, and **b** dual slope ADC

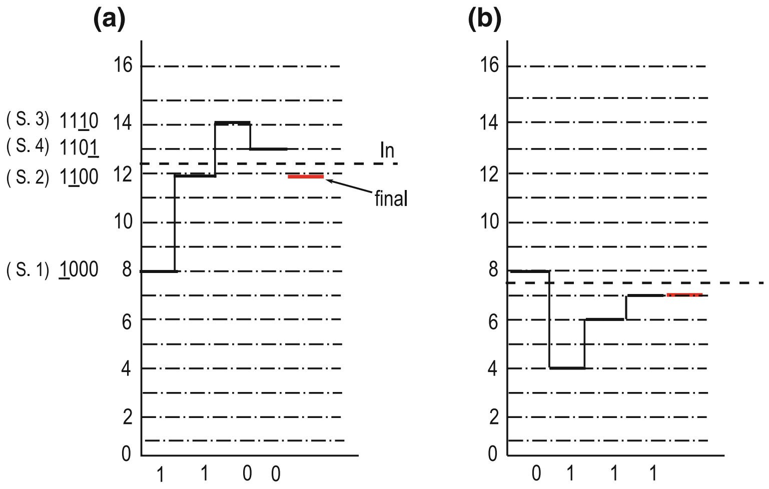


Fig. 10.40 Two 4-bit examples illustrating successive approximation

steps by comparing with mid levels of successive region subintervals. This principle is illustrated with the following example.

Example 10.13 *We illustrate the successive approximation for* 4*-bit words* $b_3b_2b_1b_0$ *using Fig. 10.40. The dotted lines show the lower bounds of the subintervals in a half LSB compensated quantization. We discuss case (a) and leave (b) to the reader.*

We first compare with the mid level 1000 *of the full scale region and by comparison in which half the input is located, defining then the most significant bit. In our example,* $b_3 = 1$ *since the input is above this level. We proceed then to compare with the mid level of the already determined region, that is, with* 1100*. Again, the input is above the level, so* $b_2 = 1$.

Hence, at this step we know $\frac{1}{4}FS$ *region where our result is, which will be of the form* 11xx. *We go on to the third step selecting the mid-point of this region* 1110 *and compare again. Since the input is now below this level,* $b_1 = 0$. *In the fourth and final step, we find that the input is below the level* 1101*, so* $b_0 = 0$. *Hence the conversion yields* 1100.

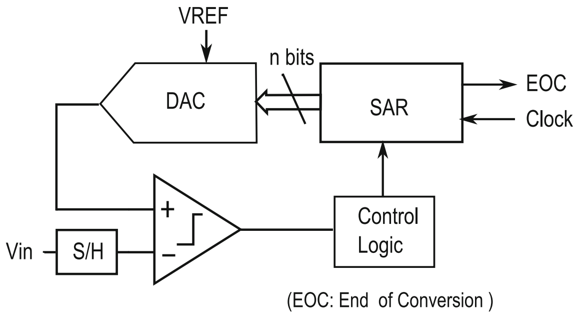


Fig. 10.41 Basic operational block diagram of a successive approximation ADC

Figure 10.41 shows the basic structure of a successive approximation ADC. The SAR block is the successive approximation register which keeps the intermediate values as well as the final one. This register feeds a DCA whose output is the one we use to compare with the input level. The process takes time, so there is a flag to indicate when the conversion has finished. The system is regulated by a clock, which may be independent of the system clock.

The main advantage of the SAR ADC is that the circuit complexity and power dissipation are less than those found in most other types of ADCs. One of the drawbacks is that eventually the comparator must do a comparison within 1 LSB of precision, and precautions must be taken to deal with noise.

Modern SAR structures use the charge distribution principles of a DAC to simplify the architecture. The charge distribution SAR ADC shown in Fig. 10.42 has the advantage that the capacitor array serves both as S/H and DAC stage. This configuration is now common in most cases in which the microcontroller has a SAR ADC peripheral. The switches are controlled by the SAR register. The total capacitance is 2C.

Conversion starts by sampling, with switch S open and all others connected to Vin. Then the hold phase opens S and all other switches connected to ground. The bits are found one by one starting with the MSB b_{n-1}.

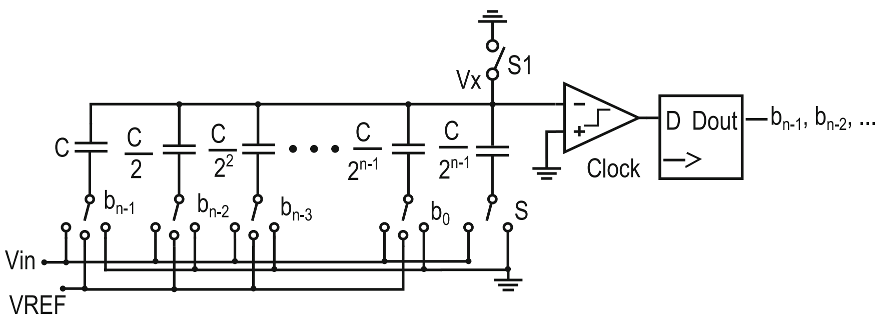


Fig. 10.42 Basic operational block diagram of a successive approximation ADC with charge distribution

As we determine the output bits, the voltage Vx is changing, until it decreases to a value within $\pm\frac{1}{2}$ LSB, when the least significant bit is determined. Remark that once bit b_j has been determined, the switch is connected to VREF if $b_j = 0$ and to ground if $b_j = 1$. The Vx value becomes, step by step,

$$Vx = -Vin \Rightarrow Vx = -Vin + \frac{\text{VREF}}{2} \Rightarrow Vx = -Vin + \frac{b_{n-1}\text{VREF}}{2} + \frac{\text{VREF}}{2^2}$$
$$\Rightarrow Vx = -Vin + \frac{b_{n-1}\text{VREF}}{2} + \frac{b_{n-2}\text{VREF}}{2^2} + \frac{\text{VREF}}{2^3} \dots$$

until it reaches its final value of

$$Vx = -Vin + \frac{\text{VREF}}{2^n}\left(b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \dots b_0\right)$$

The circuit in Fig. 10.42 is one among different configurations based on the same ADC topology and principles. Variants of it include having two reference voltages to deal with bipolar inputs and two's complement representations, input on the top instead of bottom, etc. Their discussion does not fall within the scope of this book.

10.9.3 Oversampled Converter: Sigma-Delta

The ADC architectures mentioned before belong to a class of converter types known as Nyquist rate converters, which work with a sampling frequency close to the Nyquist rate, that is $f_s \approx f_N$. Moreover, they work with quantization levels which are associated in a one to one relationship with the n-bit words. These converters are generally well suited for moderate resolutions in a high input bandwidth environment.

Nyquist rate converters have, however, two main disadvantages. One is the need of anti aliasing filters. The other is the level of quantization noise, which imposes severe constraints on the resolution and on the voltage references required, which may go to the microvolt range for large values of n. These disadvantages are addressed with *oversampling AD converters* for which $f_s > f_N$. These converters perform an inverse trade-off. That is, they allow high resolution but input bandwidths are not as high. The most popular oversampling converter is the *Sigma-Delta ADC* ($\Sigma\Delta$ ADC), sometimes also called *Delta-Sigma ADC*.[8]

The theoretical bases for this architecture go back to the 1940s, when the delta modulation concept was developed for PCM communication. The sigma-delta concept appeared in the 1950s, and the development of VLSI technology provided the definite impulse for this architecture. Because of its several advantages, it is one of the preferred methods for analog-to-digital conversion. Among them, the relaxation—or even no need—of requirements on anti aliasing filters, better management of noise, and high resolution. While 14 bits is in general a forced upper

[8] In fact, delta-sigma was first used, until 1970 when ATT engineers started using sigma-delta.

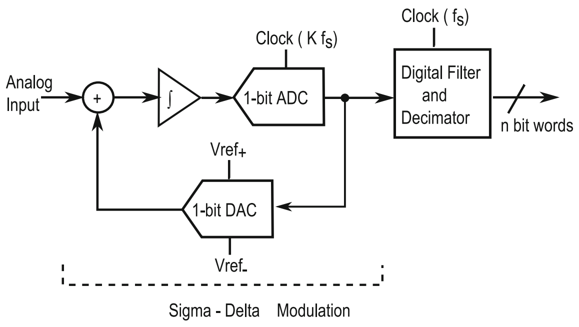


Fig. 10.43 Block diagram of a first order sigma delta ADC

bound for SAR ADCs, ΣΔ ADCs can achieve resolutions well beyond 20 bits. On the downside, the speed is in general lower than those architectures, due to the fact that they emphasize changes and not absolute values. Therefore, signals with fast changes are not easily handled.

The common architectures are the first order and second order structures of Figs. 10.43 and 10.44. Theoretically, the 1-bit ADC/DAC may be of *m* bits instead. However, for all applications and practical implementations it has been found, and theoretically proved, that 1 bit is very good.

The 1-bit ADC may be as simple as a comparator, and the 1-bit DAC an inverter, or a comparator again. The ΣΔ modulation portion works at a frequency much higher that the Nyquist frequency. The *Oversample Ratio* (OSR) is defined as

$$\mathrm{OSR} = \frac{K f_s}{f_N}$$

Frequency f_s, which controls the digital filter is close to the Nyquist frequency.

The bit stream delivered by the sigma-delta modulation subsystem can be regarded either as a digital or an analog signal. This bit stream is a one-bit serial signal with a

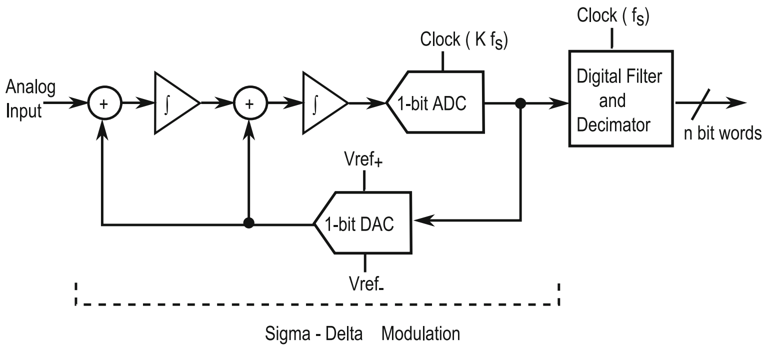


Fig. 10.44 Block diagram of a second order sigma delta ADC

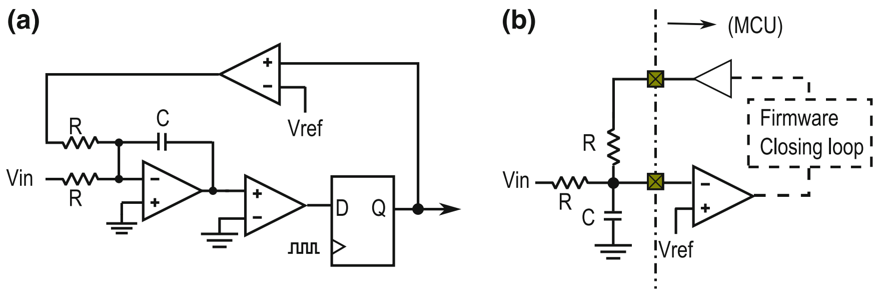


Fig. 10.45 Two examples for delta modulation block: with OA integrator. (**a**) and comparator (**b**)

very high bit rate. Its major property is that its average level represents the average input signal level, which is extracted by the filter part. In fact, this filter is a low-pass one, and it does not need to be really digital. However, working it this way has the advantage that it can be implemented by software.

One interesting advantage of sigma-delta converters is that, since the input value is determined as an average value, it is not necessary to use extremely precise element values, like in the slope ADCs. Moreover, circuit approximations are valid, as illustrated in the following example.

Example 10.14 *Figure 10.45 shows two simple realization principles of a $\Sigma\Delta$ converter. The first one uses three OpAmps, two of them as comparators. One comparator is used as a single bit ADC, the other as a DAC. In the second one, the sum and integration is done only with passive components.*[9]

In the first configuration, an MSP430 with three OAs can be used. The external elements would be the resistors and the capacitor. Ditto for the right configuration, in which case the loop may be closed with software code.

These examples provide an orientation of how we can realize a sigma delta converter for microcontrollers which do not have such peripheral.

10.9.4 Closing Remarks

We discussed briefly three types of ADCs, and mentioned the flash converter ADC too. Another interesting architecture that we are leaving out is the pipeline ADC. More architectures are available in the literature, but commercially these five are the most popular in IC form, while the first three are preferred for built in ADCs in microcontrollers, and also relatively easy to build within an MCU with only comparator included. Let us close our visit to this topic with some concluding remarks.

[9] Even for moderate clock frequencies, the circuit behaves like a summing integrator when the product CR is large enough.

Flash ADC It is the fastest of all configurations. The choice if you need very high speed and power is not of concern. It is limited however to low resolutions, 8 bits at most. Conversion time is independent of resolution.

Pipeline ADC Not discussed here, it is a good alternative for high speed applications, in the order of several Msps (Mega samples/s) to approximately 100 Msps. It offers better resolutions than the flash architecture with less power consumption.

Successive Approximation ADC It is good in the medium range of resolution, like the pipeline, but with low power consumption. The speed, however, is limited to around 5 Msps, and diminishes as resolution goes higher. It needs anti aliasing.

Sigma delta ADC It offers the highest resolution, from 16 to 26 bits. But it is slower than the successive approximation type. An advantage is that it does not require precision components and anti aliasing is not necessary or is very loose.

Slope ADC It is the slowest type. It is good for monitoring DC signals, and it consumes little power. It also exhibits very good noise performance. Depending on the realization, though, it may require high precision elements.

10.9.5 Using the MSP430 ADCs

ADCs built-in in MSP430 microcontrollers are all SAR or Sigma Delta type converters. Some models are mentioned as having slope ADC incorporated. In fact, they have comparators and I/O ports with good performance by design for these applications. Low voltage series, on the other hand, have a set of analog peripherals which allow configuring 8-bit SAR ADC converters, but do not have the converter as a built in peripheral.

The actual ADCs that are available are the 10-bit SAR ADC converter, ADC10, 12-bit SAR ADC converter, ADC12, and the 16-bit and 24-bit sigma-delta converters SD16, SD16_A, SD24, and SD24_A.

MSP430 ADC10 (ADC12) Features

ADC10 and ADC12 differ in very few aspects: resolution and management of results. Hence, we limit our introduction to the ADC10 case. A simplified block structure of the ADC10 system is shown in Fig. 10.46. We have omitted several intermediate blocks and the control signals, introduced below.

The ADC10 system has eight registers associated to it:

- Two 16-bit read-and-write control registers: ADC10CTL0 and ADC10CTL1. Some specific bits are read-only. They are reset on POR.
- Two 8-bit read-and-write input enable registers: ADC10AE0 and ADC10AE1. They are reset on POR.
- One 16-bit read-only register, ADC10MEM, to receive conversion results.

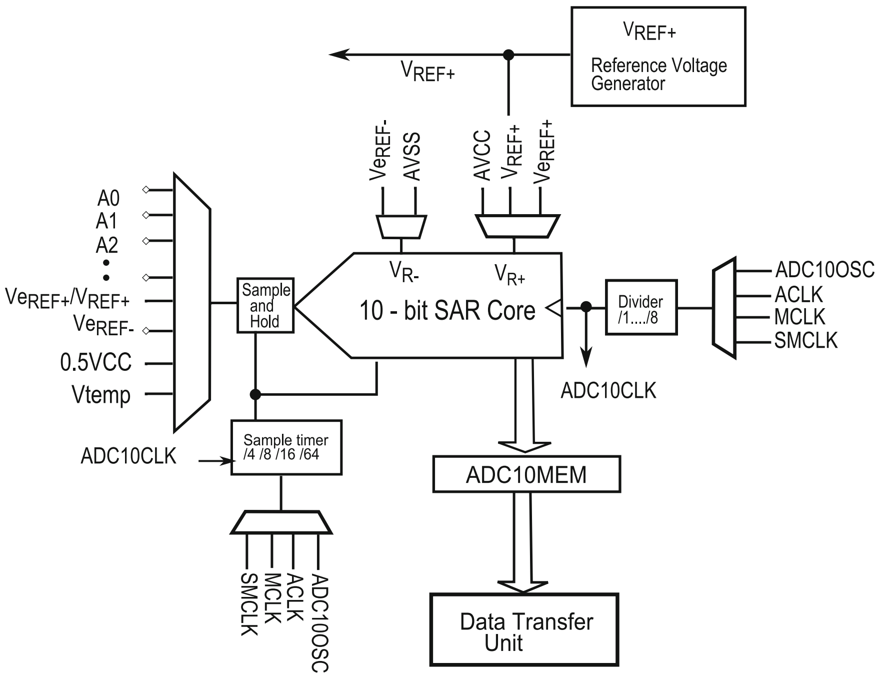


Fig. 10.46 Simplified block configuration of the MSP430 10-bit ADC10 SAR converter

- Two read-and-write data transfer control registers: ADC10DTC0 and ADC10 DTC1. Some specific bits are read-only. They are reset on POR.
- On 16-bit read-and-write data transfer start address register: ADC10SA, initialized with 0200h on POR. Bit 0 is hardwired to 0.

With reference to Fig. 10.46, we have the following:

Core The core of the system is a switched capacitor SAR converter. The SH unit in the figure is only functional, since it is practically part of the core (see Fig. 10.42). It may be turned on or off with the ADC10ON bit—Bit 4 in ADC10CTL0–. A read-only flag ADC10BUSY—bit 0 in ADC10CTL1—is set to indicate that sampling and conversion are in progress. The result is written to register ADC10MEM in a format selected with ADC10DF—bit 9 in ADC10CTL1:

- The default right justified straight binary in ADC10MEM. Bits b15 to b10 are 0. Result 0000h corresponds to the bottom of input range VR−. Result goes from 0000h to 03FFh.
- Left justified two's complement format, with bits b5 to b0 equal to 0. Result 0000h corresponds to the midpoint of input range. Results read from 8000h to 7FC0h.

The full scale is VR+ to VR−. When ADC10MEM is updated, it sets the interrupt flag ADC10IF (bit 2 in ADC10CTL0) which will generate an interrupt request if ADC10IE (bit 3) is set.

Core References VR+ may be chosen from two external references, AVCC or a special $\mathrm{Ve}_{\mathrm{REF}+}$, or else from an internally generated reference $\mathrm{V}_{\mathrm{REF}+}$. The external reference can be buffered or unbuffered. VR− may be chosen from two external references, the default AVSS or a special $\mathrm{Ve}_{\mathrm{REF}-}$. Selection of reference is done with bits 15 to 13 from ADC10CTL0, SREFx.

Internal Reference $\mathrm{V}_{\mathrm{REF}+}$ is generated internally when REFON (bit 5 in ADC10 CTL0) is set. It is by default 1.5 V, unless bit 6 REF2_5 is set, making it 2.5 V. It is very stable. This voltage is usually available at a pin if REFOUT is set, and can be buffered. The buffer is controlled with ADC10SR.

Core Clock The clock for the SAR core is a signal ADC10CLK, which is generated from a divider. This divider receives an input from one of four sources: The MSP430 clocks ACLK, MCLK, or SMCLK, or else a built-in oscillator ADC10OSC. The selection is done with bits 4-3, ADC10SSELx, and division with bits ADC10DIVx, bits 7-5 all found in control register 1.

Conversion ADC10 must be enabled with bit ENC (bit 1 in control register 0) for a conversion to take place. Conversion may be starte by software setting bit ADC10SC, or else by any of time A outputs, Timer_A.OUT0, Timer_A.OUT1 or Timer_A.OUT2. The trigger source may be selected with bits SHSx in ADC10CTL1. Once triggered, the sample and hold time is controlled with bits ADC10SHTx of ADC10CTL0, to be 4, 8, 16, or 64 times ADC10CLK. ADC10SC is automatically reset when conversion is terminated.

Inputs The analog inputs to the ADC unit may be external inputs A0, A1, …, the number depending on the model. The external references and internal reference may serve as inputs. Internally generated values 0.5 VCC or from temperature sensor may also be inputs to the converter. The selection is done with bits 15-12, INCHx, from control register 1. In the same register, bits 2-1, CONSEQx, allow to control for single or multiple conversions from different channels. Inputs A1, A2, …are shared with I/O pins. Hence, the ADC10AE0 and ADC10AE1 registers are used for enabling the connections for these pins to the converter.

Result and transfer The ADC10MEM is readable. However, we usually need to transfer the results to another place in memory. This is done with the ADC10 data transfer controllers. One convenient feature is that data transfer is done without the CPU intervention (it may even be off) directly interacting with DMA. These registers function as follows:

- ADC10DTC0: selects one or two blocks in destination, or continuous transfers.
- ADC10DTC1: provides the number of transfers per block. Transfer is disabled when the value is 0.
- ADC10SA is the start address for destination. To enable transfers, the user must write to this register.

Let us work some simple code examples to set up the ADC10 and work conversions.

Example 10.15 *Let us set up ADC10 to work with input* A0 *(pin shared with* P2.0*). We use the converter oscillator* ADC10OSC *(default) for generation of signal* ADC10CLK*, the Sample and hold time will be eight times this signal. Straight binary conversion is performed. The CPU is in low power mode while the ADC10 is working. The following piece of code works for our purpose:*

```
;===================================================================
; ADC10SHT_1 for 8x ADC10CLKs and ADC10IE to enable interrupt

setup_ADC   mov.w   #ADC10SHT_1+ADC10ON+ADC10IE,&ADC10CTL0 Input_A0
bis.b   #01h,&ADC10AEO           ; select A0 for input Setup_P2
bis.b   #01h,&P2SEL              ; pin P2.0 to ADC10 ; Mainloop
bis.w   #ENC+ADC10SC,&ADC10CTL0 ; Start sampling
                                           ; and conversion
            bis.w   #CPUOFF+GIE,SR          ; LPM0
            jmp     Mainloop                ; Again

;--------------------------------------------------------------------
; - - - - - - - - - - - - - - -
ADC10_ISR
           bic.w   #CPUOFF,0(SP)            ; CPU Active on return

           -   -   -   -                    ; ISR instructions
           reti                             ; return from interrupt

;---------------------------------------------------------------------
; - - - Interrupt vector  -  -

           ORG    ADC10_VECTOR
           DW     ADC10_ISR
;=====================================================================
```

A C-code is the following:

```
//==================================================================
void main(void)
{
  // 8 times ADC10CLKs, ADC10ON, interrupt enabled
  ADC10CTL0 = ADC10SHT_1 + ADC10ON + ADC10IE;
  ADC10AE0 |= 0x01;                  // P2.0 ADC A0 option select
  for (;;)
  {
    ADC10CTL0 |= ENC + ADC10SC;      // Sampling & conv. start
    __bis_SR_register(CPUOFF + GIE); // LPM0,
  }
}

//------------------------------------------------------------------
// ADC10 interrupt service routine
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
 __bic_SR_register_on_exit(CPUOFF);  // Active on Return

  - - - - - - - - -                   // ISR instructions here
}
//==================================================================
```

These code examples are limited to setting of the ADC10.

MSP430 SD16_A Sigma-Delta Converter

The MSP430 has three variations for its sigma-delta ADC. The original one SD16 contains three independent channels, each one being a complete ADC with a single input. They can operate simultaneously or with specified delay in between, and are linked with logic.

The SD16_A has a single core but several inputs which may be sequentially but not simultaneously converted. It also has enhancements with respect to the previous model. A third module is an extension of this model containing several single input SD16_A cores linked with logic.

Figure 10.47 shows a block diagram of the SD16_A module, showing the control bits for different blocks. The converter has three registers associated to the core: SD16_A control register SD16CTL, SD16_A interrupt vector register SD16IV, and the SD16_A analog enable, SD16AE. Each channel x has additional registers for control, conversion and input control, SD16CCTLx, SD16MEMx, and SD16INTCTLx, where x stands for the channel number. Series 'x4xx models have also a preload register SD16PREx.

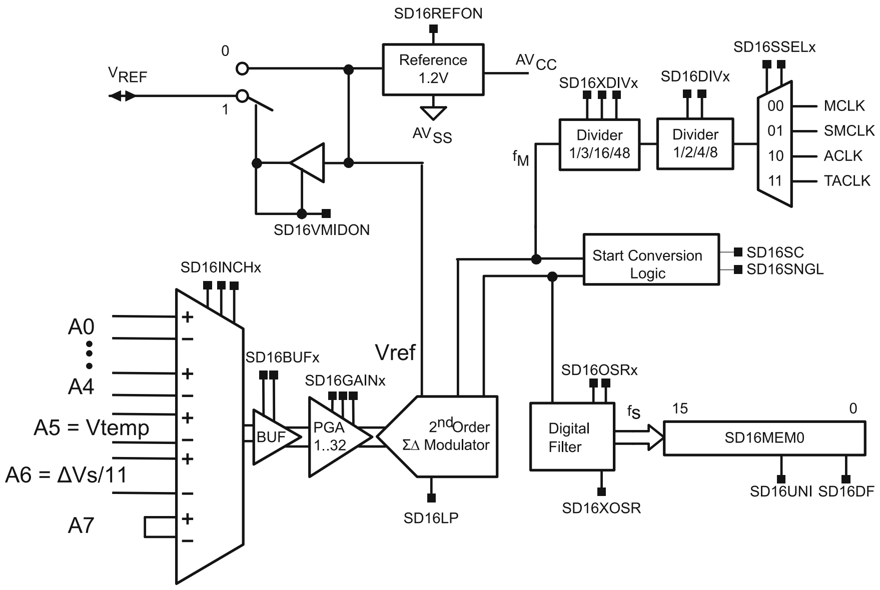


Fig. 10.47 Simplified block configuration of the MSP430 Sigma-Delta converter SD16_A (*Adapted from MSP430 user guide, Texas Instruments Inc.*)

The reference voltage V_{ref} for the converter can be externally supplied or generated internally with a 1.2 V reference voltage generator when REFON is set. This voltage is also buffered to be externally available.

The clock, on the other hand, is selected from one of the MSP430 signals MCLK, SMCLK, ACLK, and TACLK and divided first by 1, 2, 4, or 8 with the SD16DIVx bits, and then by 1, 3, 16, or 48 with the SD16XDIVx bits. This will generate the modulator frequency signal f_M. The outputs from the filter are generated at a frequency

$$f_S = f_M/OSR$$

The oversampling ratio OSR is set by the SD16OSRx bits as 32, 64, 128 and 256. SD16XOSR bit can give increased ratios of 512 or 1024. Unlike the ADC10 and ADC12, the core is automatically shut down when not in used.

The system has eight differential inputs selectable with the SD16INCHx bits. Input A5 is internally generated by a voltage divider yielding $\Delta Vs = (VCC - VSS)/11$, a useful feature to monitor state of battery. Input A6 is generated by the temperature sensor, and A7 has the inputs short circuited, which is useful for calibration.

The inputs go through a high impedance differential buffer, not available in all models, whose current capacity is configured with SD16BUFx. The buffer is convenient for low impedance inputs. This block is followed by a fully differential programmable gain amplifier configured with SD16GAINx bits. Gains are from 1 to 32 in powers of 2. The differential input voltage should be between $\pm V_{FSD}$, where

$$V_{FSD} = \frac{0.5 V_{ref}}{PGAGain} \tag{10.47}$$

conversion starts when the SD16SC bit is set. It may work a single conversion or the default continuous conversions depending on SD16SNGL bit. To fully realize a single conversion, the core has to perform a number of conversions which is specified by the SD16INTDLYx bits. On the other hand, continuous conversions do not stop until SD16SC is cleared by software, in which case conversions stop without completing the current one.

The interrupt flag SD16IF is set when a new result is available at register SD16MEM0. The flag is automatically reset when the result is read. Multiple channels version have one register and one interrupt flag for each channel. An overflow interrupt flag SD16OVIFG is set when a new result overruns a previous value which had not been read. These flags generate maskable interrupt requests when SD16IE is set.

The output format can be configured using the SD16DF and SD16UNI bits in anyone of the following: (a) bipolar two's complement, (b) bipolar offset binary, and (c) unipolar binary. These output formats are illustrated in Fig. 10.48. Remember that the upper bound of the full scale is not encoded. This is shown with parenthesis.

The following two listings, courtesy of Texas Instruments, Inc., show examples using this ADC converter. The first one is in assembly language and the second one

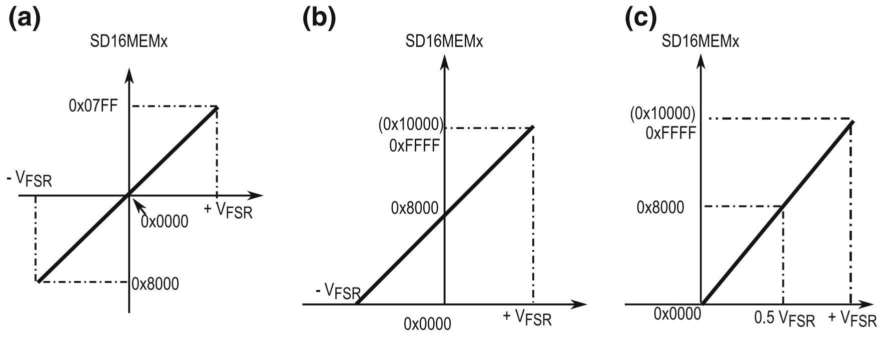


Fig. 10.48 MSP430 Sigma-Delta converter SD16_A output formats: **a** bipolar two's complement, **b** bipolar offset binary, and **c** unipolar binary (*Courtesy of Texas Instruments, Inc.*)

in C language. In the latter case, the documentation is omitted for brevity since it contains the same information.

Example 10.16 *This example program uses the SD16 module to perform a single A-to-D conversion on Channel 2. A interrupt from the SD16 is configured to occur upon end-of-conversion.*

To test the program, apply a known voltage at CH2 (A2.0+, A2.0−) and insert a breakpoint at the line indicated in the code comments. The converted value can be examined in R12 with the debugger.

The assumed settings include $ACLK = LFXT1 = 32768\,Hz$ *and* $MCLK = SMCLK = \text{default } DCO = 32 \times ACLK = 1048576\,Hz$*. An external clock crystal is required for ACLK and a 100 nF capacitor between Vref and AVss is recommended when using* $V_{ref} = 1.2\,V$.

```
;=============================================================================
; Demo - SD16, Single Conversion on Single Channel Using ISR
;
; By H. Grewal * Texas Instruments, Inc. *  February 2005
; Built with IAR Embedded Workbench Version: 3.21A
;=============================================================================
#include  <msp430x42x.h>
;- - - - - - CPU Registers Used - - - - - - - - - - - - - - - - - -
#define     RESULTS R12      ; R12 - Store conversion result from SD16MEM2
                             ; R15 - Temporary working register

;-----------------------------------------------------------------------------
            ORG     08000h                  ; Program Start
;-----------------------------------------------------------------------------
RESET       mov.w   #600h,SP                ; Initialize SP
StopWDT     mov.w   #WDTPW+WDTHOLD,&WDTCTL  ; Stop watchdog
SetupFLL    bis.b   #XCAP14PF,&FLL_CTL0     ; Configure load caps
            mov.w   #10000,R15              ;
Xtal_Wait   dec.w   R15                     ; Delay for 32 kHz crystal to
            jnz     Xtal_Wait          ; stabilize
SetupSD16   mov.w   #SD16REFON+SD16SSEL0,&SD16CTL
                                            ; 1.2V ref, SMCLK
            bis.w   #SD16SNGL+SD16IE,&SD16CCTL2
                                        ; Single conv, enable interrupt
            bis.b   #SD16INTDLY0,&SD16INCTL2
                                        ; 3rd sample generates interrupt
            mov.w   #03600h,R15         ; Delay  for 1.2V ref startup
L$1         dec.w   R15             ;
            jnz     L$1             ;
            eint                    ; Enable general interrupts
Mainloop    bis.w   #SD16SC,&SD16CCTL2  ; Start conversion
            bis.w   #CPUOFF,SR      ; Enter LPM0 (disable CPU),
                                    ; wait for conversion to complete
            nop                     ; Required for debug only
            jmp     Mainloop        ; SET BREAKPOINT HERE

;-----------------------------------------------------------------------------
SD16_ISR    ; SD16 Interrupt Service Routine
;-----------------------------------------------------------------------------
            add.w   &SD16IV,PC          ; Add offset to PC
            reti                        ; Vector 0: No interrupt
            jmp     SD_OV               ; Vector 2: Overflow
            jmp     SD_CH0              ; Vector 4: CH0 IFG
            jmp     SD_CH1              ; Vector 6: CH1 IFG
                                        ; Vector 8: CH2 IFG

;-----------------------------------------------------------------------------
;- - - - - -SD16 Channel 2 Interrupt Handler;- - - - - - - - -
SD_CH2      mov.w   &SD16MEM2,RESULTS   ; Save CH2 conversion
            bic.w   #CPUOFF,0(SP)       ; Return active
SD_CH2_END  reti                        ; Return from interrupt

;-----------------------------------------------------------------------------
;- - - - - -SD16 Memory Overflow Interrupt Handler - - - - - - SD_OV
reti                        ; Return from interrupt

;-----------------------------------------------------------------------------
;- - - - - -SD16 Channel 0 Interrupt Handler- - - - - - - - - -
SD_CH0      reti                        ; Return from interrupt

;-----------------------------------------------------------------------------
;- - - - - -SD16 Channel 1 Interrupt Handler- - - - - - - - - -
SD_CH1      reti                        ; Return from interrupt

;-----------------------------------------------------------------------------
            COMMON  INTVEC              ; Interrupt Vectors
;-----------------------------------------------------------------------------
            ORG     RESET_VECTOR        ; RESET Vector
            DW      RESET               ;
            ORG     SD16_VECTOR         ; SD16 Vector
            DW      SD16_ISR            ;
            END
;=============================================================================
```

Example 10.17 *A C-language version of the program in the previous example.*

```
//============================================================================
//  Demo - SD16, Single Conversion on Single Channel Using ISR
//----------------------------------------------------------------------------
//  By H. Grewal * Texas Instruments, Inc. *  February 2005
//  Built with IAR Embedded Workbench Version: 3.21A
```

```
//==========================================================================
#include  <msp430x42x.h>
//--------------------------------------------------------------------------
unsigned int result;

void main(void)
{
  volatile unsigned int i;          // Use volatile to prevent removal
                                    // by compiler optimization

  WDTCTL = WDTPW + WDTHOLD;         // Stop WDT
  FLL_CTL0 |= XCAP14PF;             // Configure load caps
  for (i = 0; i < 10000; i++);      // Delay for 32 kHz crystal to
                                    // stabilize

  SD16CTL = SD16REFON+SD16SSEL0;    // 1.2V ref, SMCLK
  SD16CCTL2 |= SD16SNGL+SD16IE ;    // Single conv, enable interrupt
  SD16INCTL2 |= SD16INTDLY0;        // Interrupt on 3rd sample
  for (i = 0; i < 0x3600; i++);     // Delay for 1.2V ref startup

  _EINT();                          // Enable general interrupts

  while (1)
  {
    SD16CCTL2 |= SD16SC;            // SET BREAKPOINT HERE
                                    // Set bit to start conversion
    _BIS_SR(LPM0_bits);             // Enter LPM0
  }
}

//--------------------------------------------------------------------------
#pragma vector=SD16_VECTOR __interrupt void SD16ISR(void) {
  switch (SD16IV)
  {
  case 2:                           // SD16MEM Overflow
    break;
  case 4:                           // SD16MEM0 IFG
    break;
  case 6:                           // SD16MEM1 IFG
    break;
  case 8:                           // SD16MEM2 IFG
    result = SD16MEM2;              // Save CH2 results (clears IFG)
    break;
  }

  _BIC_SR_IRQ(LPM0_bits);           // Exit LPM0
}
//==========================================================================
```

10.10 Chapter Summary

This chapter described the basic components of the analog signal chain. That is, the components that process the analog signal prior to the digital processor, or that extract analog information from the digital processor's output.

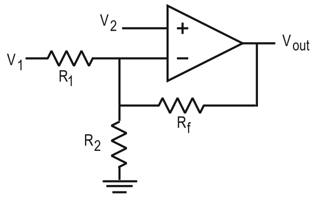


Fig. 10.49 A subtractor circuit

The chain starts with a sensor, which transforms a non-electrical signal into one that can be processed by electronic means. An overview of sensors was covered in this chapter.

The sensor's output may need conditioning so that it can satisfy the requirements of the electronic components that process the signal. This conditioning may include amplification, filtering, or other pre-processing. This stage uses operational amplifiers, another of the signal chain components. Different circuits were discussed for this stage.

The conditioned signal is then sampled and converted into a digital word using an Analog-to-Digital converter (ADC). The theoretical principles behind this transformation, as well as the basic architectures were reviewed.

The conversion from the digital output to an analog value was similarly considered. The Digital-to-Analog converter (DAC) was reviewed.

Finally, a tour on the embedded analog signal chain peripherals in the MSP430 devices was introduced, so the reader can gain familiarity with these important components of the MSP430 architecture.

10.11 Problems

10.1 In general, human speech generates frequencies from 50 Hz upward, to about 10 kHz. However, most energy is concentrated in the 300 Hz–3.4 kHz band, which is the one in which speech intelligibility is the best. On this basis, the telephone system and many other speech recognition systems sample the speech at a rate of 8 kHz. Justify the use of this frequency.

10.2 Starting with Eq. 10.6, with $V_r = 0$, find the output voltage expression for the subtractor when $R_4/R_3 \neq R_2/R_1$.

10.3 The circuit in Fig. 10.49 is another example of a subtractor circuit. It is often used to provide amplification to a sensor signal V_1 and an offset (*scaling*) to shift toward a desired origin or to keep the output voltage in a linear region, especially in cases of one supply OAs. The circuit can also be considered as a non-inverting amplifier with shifting. Find the expression for the output voltage.

10.4 With a resolution of N bits and using as quantization levels those in Eq. (10.24), what is the quantization error for Vmax?

10.5 Show that in an N-bit quantization for symmetrical intervals, if N_{QB} is the representation in biased encoding and N_{QT} in two's complement encoding, then $N_{QB} - N_{QT} = 2^{N-1}$.

10.6 The Texas Instruments' TLV5604 is a 10-bit four channel serial digital-to-analog converter (visit http://www.ti.com/lit/ds/symlink/tlv5604.pdf for full reference). Sketch an appropriate interface connection with your MSP430 microcontroller and design a flowchart to deliver the digital information to the DAC.

10.7 The AD9760 (data sheet at http://www.analog.com/static/importedfiles/data_sheets/AD9760.pdf) is a 10-bit parallel DAC available from Analog Devices. This converter has a current output. Sketch an appropriate interface connection with your MSP430 microcontroller and design a flowchart to deliver the digital information to the DAC.

References

1. Bryant, J., & Kester, W. (2011). *Ch. 3 DATA CONVERTER ARCHITECTURES: Sec. 3.1 DAC ARCHITECTURES*. Analog Devices: Inc