# AFE implementation using PSOC™ Automotive 4 MCU

## About this document

### Scope and purpose

This application note explains how to use PSOC™ Automotive 4 MCU to implement analog front ends (AFEs) in sensing applications. It discusses commonly used AFE implementations for measurement of three types of sensors with voltage, resistance, and capacitance outputs.

### Intended audience

This application note is intended for engineers who wants to design analog front ends (AFE) for their application and those with experience in working with PSOC™ Automotive 4 microcontroller and ModusToolbox™ IDE.

# Table of contents

**About this document** . . . 1
**Table of contents** . . . 2
**1 Introduction** . . . 3
**2 AFE for different sensor types** . . . 4
**3 AFE for voltage output sensor** . . . 5
3.1 Design implementation . . . 5
**4 AFE for resistive sensor** . . . 7
4.1 Current source method . . . 7
4.2 Resistor divider method . . . 8
4.2.1 Code implementation for resistor divider method . . . 9
4.3 Accuracy analysis . . . 13
4.3.1 Tolerance of reference resistor . . . 14
4.3.2 ADC offset and gain errors . . . 14
4.3.3 ADC nonlinearity error . . . 15
4.3.4 Error in physical quantity (temperature or pressure) . . . 15
4.3.4.1 Reference resistor selection . . . 16
4.3.4.2 Conversion accuracy . . . 16
**5 AFE for capacitive sensor** . . . 17
5.1 Single slope method . . . 17
5.2 Sigma delta modulator method . . . 18
5.3 Measurement using multi-sense converter (MSC) peripheral . . . 20
5.4 Impact of parasitic capacitance . . . 22
5.5 Design implementation for capacitance measurement . . . 22
5.6 Accuracy analysis . . . 23
**6 Summary** . . . 24
**References** . . . 25
**Glossary** . . . 26
**Revision history** . . . 27
**Trademarks** . . . 28
**Disclaimer** . . . 29

# 1 Introduction

PSOC™ Automotive 4 MCU simplifies the design of sensor-based systems by delivering a scalable and reconfigurable architecture that integrates programmable analog front ends (AFEs) to a signal processing engine. PSOC™ Automotive 4 MCU enables designs to send aggregated, pre-processed, and formatted sensor data over serial communication interfaces to host processors.

Analog sensors generally come in five different types, depending on their output electrical signal, such as voltage, current, resistance, capacitance, or inductance. Each sensor type requires a specific AFE design. For example:

- A thermocouple, which is a voltage-output temperature sensor, requires an instrumentation amplifier
- An ambient light sensor, which is a current-output sensor, requires a transimpedance amplifier (TIA)

Systems that use multiple analog sensors usually require multiple specialized ICs to implement the AFE, which increases bill of materials (BOM) cost and PCB size. Systems designed for automotive applications must combine data from multiple sensors to enable new sensing capabilities, commonly known as "sensor fusion". Sensor fusion solutions often require custom AFEs. PSOC™ Automotive 4 MCU reduces the need for specialized ICs, offering the ability to create custom AFEs in a single-chip solution.

Figure 1 shows a generic block diagram of a sensor-based system. It includes:

- An AFE to condition the sensor outputs by amplifying and filtering the signals
- An analog-to-digital converter (ADC) or a comparator (not shown) to convert the conditioned sensor output into digital data
- A programmable signal processing engine with a serial communication interface to format the sensor data and send it to the host processor. A signal processing engine (32-bit Arm® Cortex®-M0+) can calibrate and tune the AFE in software

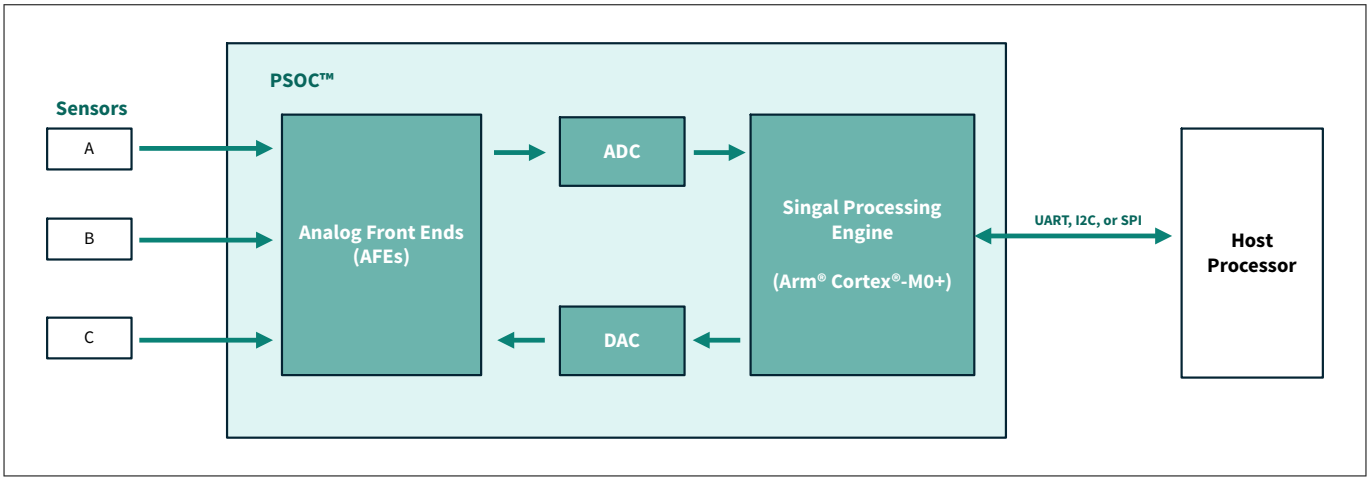



**Figure 1 Sensor-based system**

This application note explains various AFE implementation methods for voltage, resistance, and capacitance output sensors. It also discusses performance analysis for each type of sensor. Additionally, it explains how to interface these sensors with PSOC™ Automotive 4 MCU using code example references.

The References section provides a set of documents to accelerate your learning. If you are new to PSOC™ 4100S Max and 4100S Plus, see AN79953 – Getting started with PSOC™ 4 MCU to understand the device and to learn how to create a simple design.

# 2 AFE for different sensor types

Analog sensors are classified into five types: resistive, inductive, capacitive, voltage, or current, depending on their output parameter. The output of these sensors change based on a physical entity. For example, in a thermistor, the resistance changes based on the temperature. Each sensor type requires the right AFE.

Although there are five types of output, the basic electrical parameters that can be measured by a microcontroller (MCU) are voltage, time, and frequency. The output of the sensor is converted to one of these three types of electrical parameters to determine the equivalent digital count. For example, in a thermistor, the resistance of the sensor changes due to temperature. The thermistor can be placed in series with a reference resistor to form a resistive divider, which is driven by a constant voltage source, and then the voltage drop across the thermistor and reference resistor can be measured. The resistance of the thermistor can be calculated from the measured voltage drop values and the known reference resistor.

Each AFE explained in this application note uses a specific method to convert a sensor output to a basic electrical parameter with voltage, time, or frequency.

# 3 AFE for voltage output sensor

In a voltage output sensor, the voltage across sensor terminals varies with respect to the physical quantity. A few examples of voltage output sensors are:

- Thermocouples, where the voltage output changes with temperature
- Hall-effect sensors, where the voltage output varies with the magnetic field caused by the current flowing in an adjacent or built-in conductor
- PIR motion sensors, where the voltage output of the sensor changes when an infrared-emitting object passes in front of the sensor

Figure 2 shows a generic block diagram of the signal chain for a voltage output sensor.

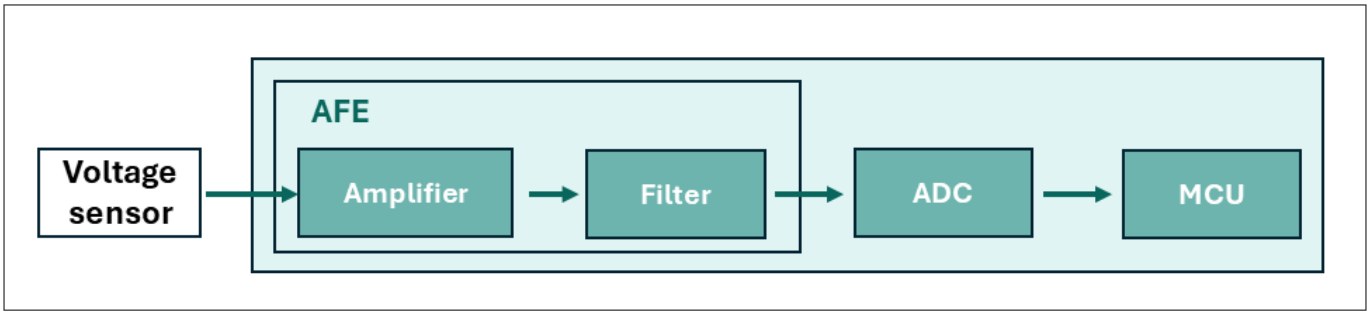



**Figure 2 Voltage measurement AFE block diagram**

The output of the sensor is primarily connected to an amplifier. The function of the amplifier is to amplify the input signal to be compatible with the ADC's input range. For example, if a voltage output of the sensor has a maximum output of 30 mV and the ADC has a range of 1 V, then the amplifier can have a gain of 32 to amplify the signal to 0.96 V, allowing a 0.04 V margin for offset voltage, signal overshoot, etc. For signals that are already in the ADC's measurement range, the amplifier can be a unity gain buffer or may not even be needed.

Many sensors require an amplifier with a high input impedance. If the sensor signal is HIGH enough to bypass the amplifier stage, the output of the sensor can be connected to an ADC through a unity gain buffer that provides HIGH input impedance to the ADC. For sensors, such as thermocouples and current shunts, a differential amplifier is preferred. For the differential sensors with LOW output impedance and significantly higher output voltage signal, the output of the sensor can directly be connected to the differential ADC. For single-ended signals such as Hall-effect sensors, a single-ended amplifier can be used.

The output of the amplifier is then connected to an optional filter. The filter can have many functions, depending on the signal being measured. Some of the functions are:

- Low-pass filter for anti-aliasing
- Notch filter to attenuate power supply noise
- Band-pass or high-pass filter to remove DC bias and extract only the interested band of the signal

The output of the filter is then connected to the ADC. The MCU can further process the digitized signal from the ADC. Note that in a resource constraint design, an MCU like CM0+ can be used to implement firmware filters.

## 3.1 Design implementation

This section provides how to implement an AFE for a PIR motion sensor, which is a voltage output sensor.

The PIR motion sensor uses infrared sensitive materials as the sensing elements. It is packaged with a field effect transistor (FET) in the source follower mode. An FET is required to buffer the high-impedance output of the sensor element.

## 3 AFE for voltage output sensor

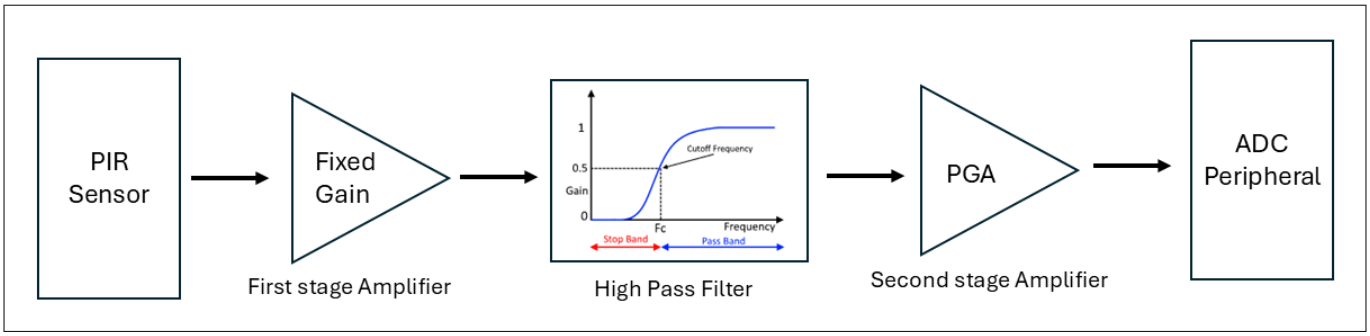



**Figure 3 Block diagram**

When the sensor element is exposed to infrared radiation, a voltage is generated across the element. These elements are arranged such that the voltage generated by one element is subtracted by the other. This arrangement cancels the common signal and generates a voltage only when there is a difference in the incident infrared radiation level on the sensing elements. The sensor package is designed to have a unique field-of-view (FoV) for each element. When an IR radiating source moves across the fields of view, the sensor generates a differential signal across the load resistor. To increase the field-of-view angle, a Fresnel lens is mounted on the PIR motion sensor. It improves the sensitivity and so the detection distance.

The sensor voltage pulse is first passed through a high pass filter to block the DC voltage from the sensor into the next stages. The peak-to-peak voltage signal from the high pass filter is on the order of millivolts, and must be amplified. To detect the motion of a human body at a distance of 10 feet, an amplifier with a gain of ~1000 is required. A single-stage amplifier with such a high gain causes the amplifier output to become saturated. Therefore, a two-stage amplifier is best suited for amplifying with such a high gain.

The total gain is split between two stages:

- The first stage amplifier uses a non-inverting amplifier configuration, using an internal operational amplifier and external gain setting resistors where the gain is fixed
- The second stage amplifier uses a programmable gain amplifier (PGA) where the gain is configurable

The internal bandgap reference voltage (BGR) is used to bias the PIR sensor and amplifier stages. The output of the second stage PGA is connected to the SAR ADC.

# 4 AFE for resistive sensor

In resistive sensors, the resistance of the sensor varies with respect to the physical quantity being measured. A few examples of resistive sensors are:

- Thermistor and resistance temperature detector (RTD) sensor, where the resistance varies with temperature
- Strain gauge, where the resistance varies with load
- Light dependent resistor (LDR), where the resistance varies with the intensity of light

This section explains two methods for resistance measurement. It also explains the implementation in PSOC™ 4100S Plus and PSOC™ 4100S Max devices with respective code examples. An Accuracy analysis section explains the errors associated with one of the resistance measurement methods, and tips to handle such errors.

## 4.1 Current source method

In this method, the sensor ($R_{sensor}$) shown in Figure 4 is excited using a constant current source. The voltage drop across the sensor is measured using an ADC. After the voltage is measured, the resistance of the sensor is calculated using (1). In this method, the accuracy of the resistance measurement depends on the accuracy of the current source, and the offset and gain errors of the ADC. Note that the ADC gain error depends on the accuracy of the reference voltage.

$$R_{sensor} = V_{sensor}/\mathrm{I} \quad (1)$$

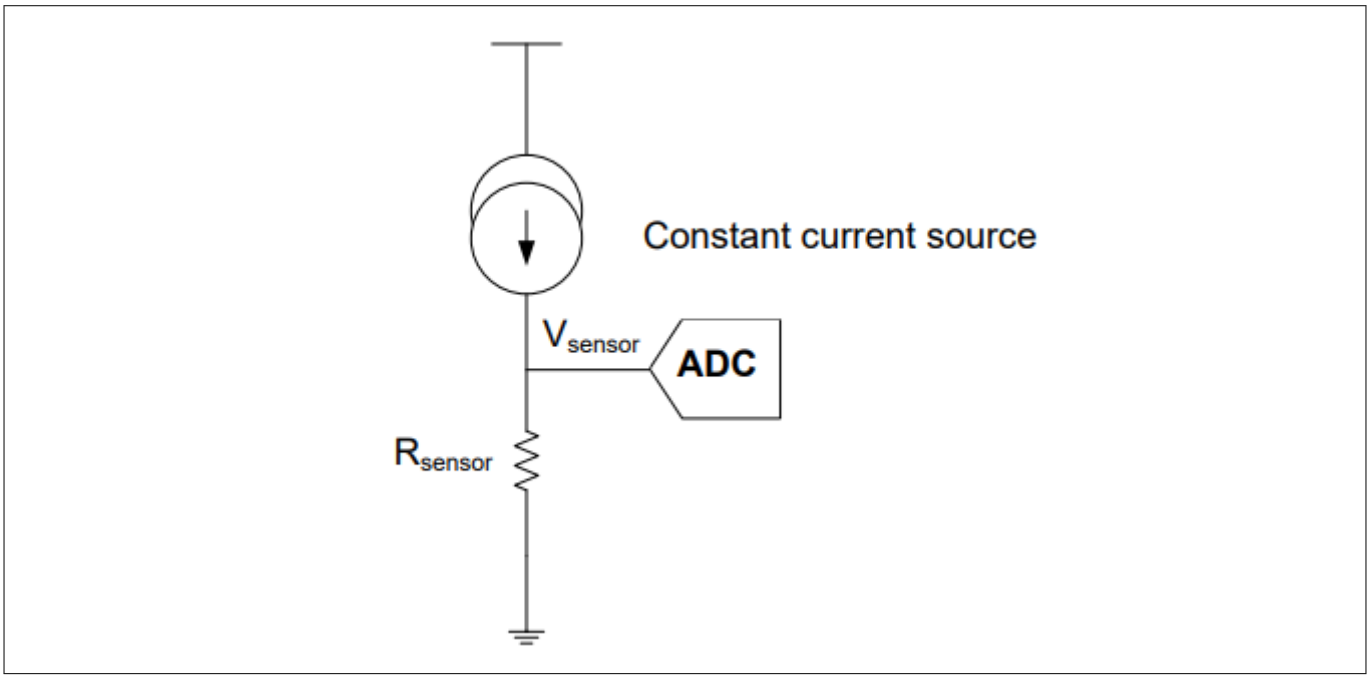



**Figure 4 Resistance measurement using a constant current source**

The following are the demerits of the current source method:

- The current source must be very accurate; any current error causes an error in the measured resistance and therefore, an error in the measured physical entity
- The offset, gain, and integral nonlinearity (INL) error of the ADC caused by an inaccurate reference voltage can lead to inaccuracies in the measured resistance
- Nonlinearity of $R_{sensor}$ will cause a nonlinearity of $V_{sensor}$, which will cause an error in the measured physical entity

## 4.2 Resistor divider method

The method shown in Figure 5 is used to reduce some of the errors associated with the previous method. A reference resistor ($R_{ref}$) and the sensor ($R_{sensor}$) form a resistor divider, which is excited using a voltage or current source. The resistor divider can also be excited from the device supply voltage based on the requirement. $R_{sensor}$ is calculated using the voltage developed across the resistor.

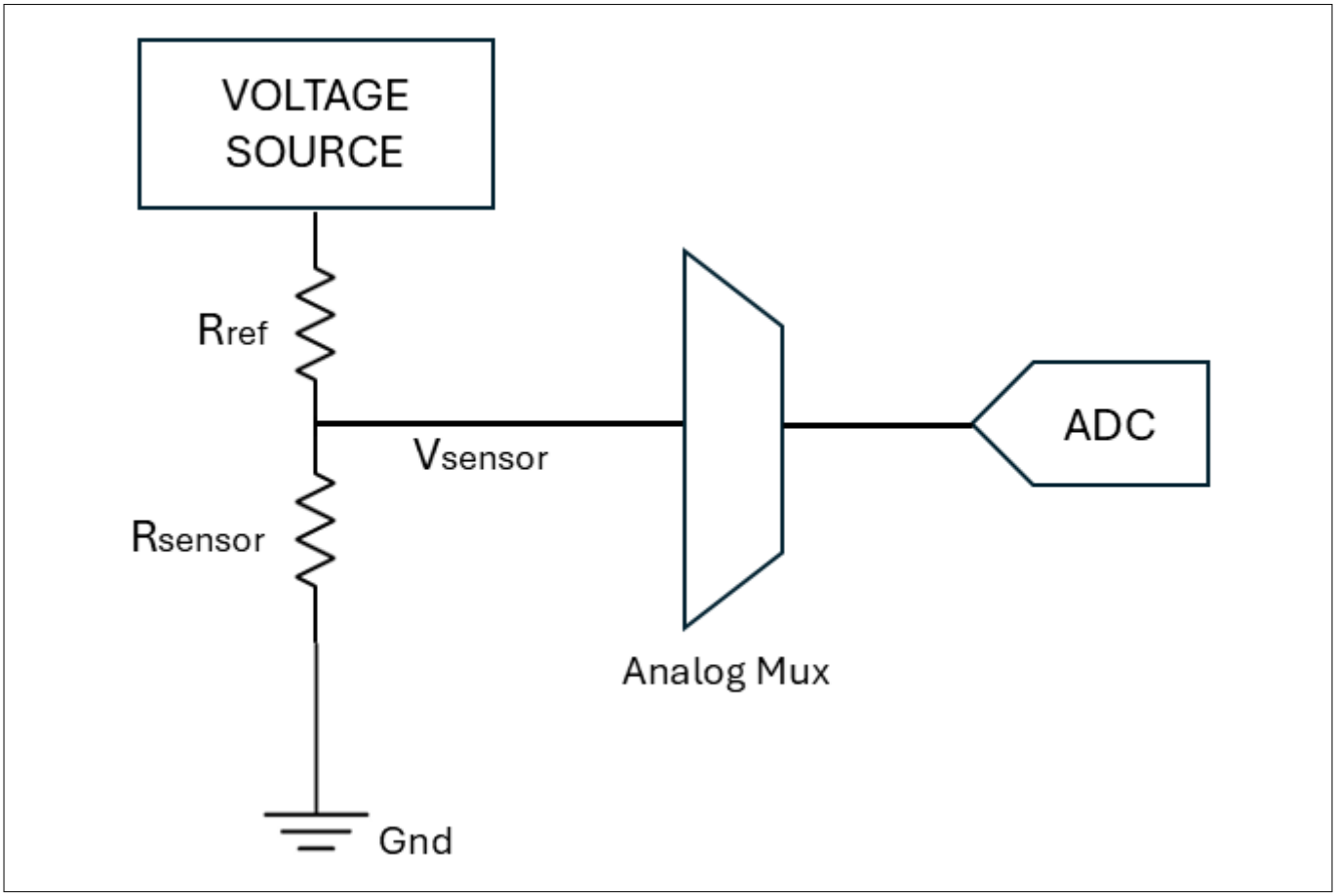



**Figure 5 Resistance measurement using a reference resistor**

***Note**: The location of the $R_{ref}$ and $R_{sensor}$ can be exchanged based on the measurement range of the physical quantity. For example, if temperature is the physical quantity being measured and if you use a negative temperature coefficient element like a thermistor, it is recommended to place the thermistor on the top and the reference resistor at the bottom. This arrangement ensures that the low resistance of a thermistor at a high temperature does not cause the ADC input signal to go below the measurement range.*

An ADC is used to measure $V_{SENSOR}$ and the sensor resistance is calculated from (2).

$$R_{sensor} = \frac{(V_{SENSOR} - V_{GND})}{(V_{SOURCE} - V_{SENSOR})} * R_{ref} \quad (2)$$

Where,

$V_{SENSOR}$ = Voltage drop across the thermistor sensor

$V_{GND}$ = Ground voltage (usually zero volts neglecting the offset errors)

$V_{SOURCE}$ = Supply voltage to the circuit

In this method, the accuracy of the sensor resistance measurement depends only on the accuracy of the reference resistor and the ADC linearity. Using a precise reference resistor is the cheapest way to improve the precision of this measurement circuit. This method is independent of the errors in the excitation source as well as the ADC offset and gain errors. In (2), the ADC's offset error gets canceled in the subtraction operations in the numerator and denominator, and the ADC's gain error gets canceled by the ratio of numerator and denominator.

### 4.2.1 Code implementation for resistor divider method

The following figures show the Device Configurator settings for the resistor divider method code implementation.

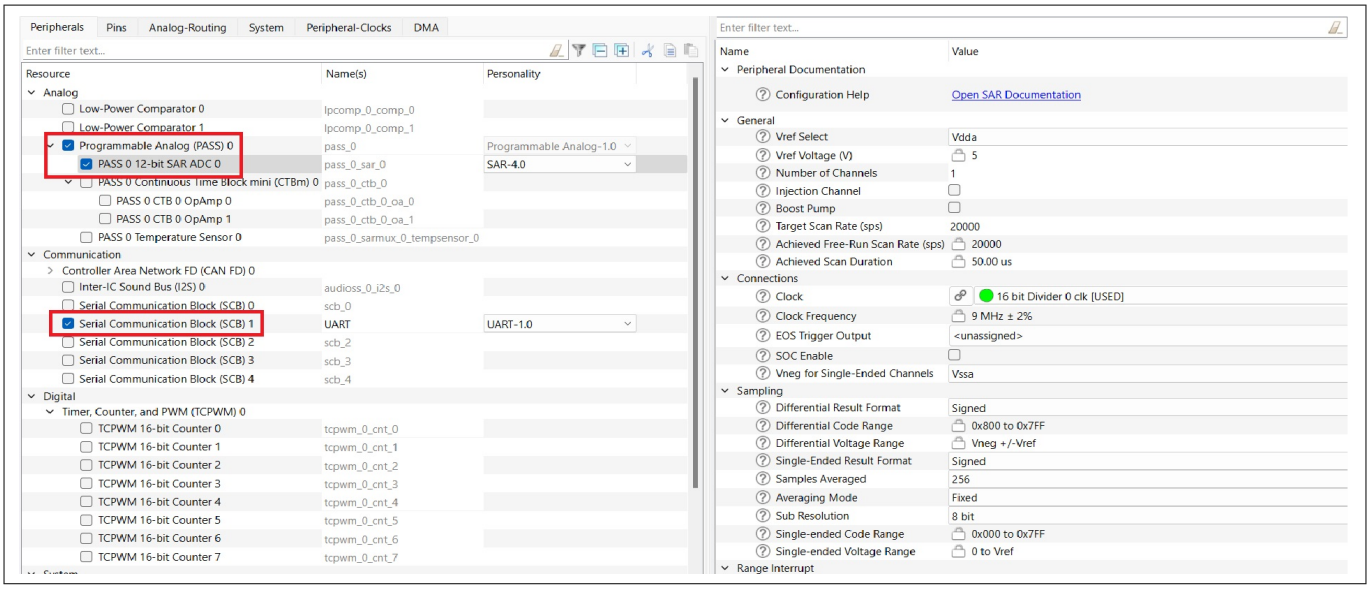



**Figure 6 Device Configurator settings**

In Figure 6, observe that the SAR ADC is configured with specific settings, and the UART SCB block is also enabled, as the temperature results are being transmitted via UART.

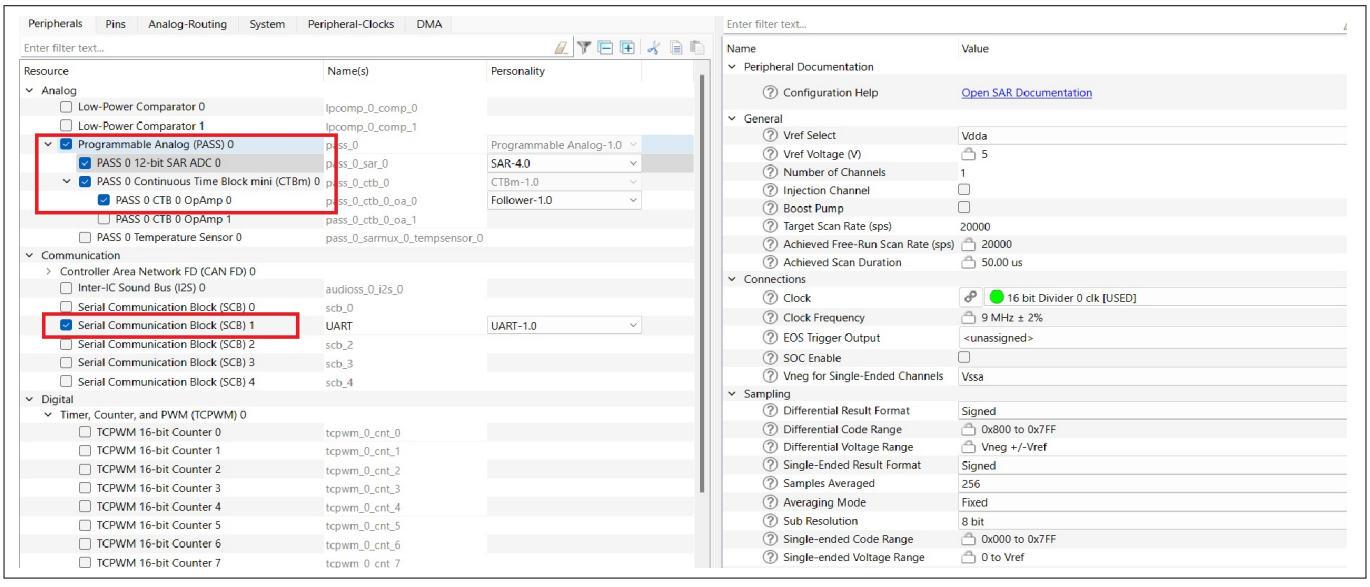



**Figure 7 Device Configurator settings**

## 4 AFE for resistive sensor

If you see the above configuration settings remains the same as before, with the addition of an enabled opamp. The sensor signal is routed to the opamp input, and its amplified output is then connected to the ADC input. This optional method can be used when the sensor's output signal is weak.

## 4 AFE for resistive sensor

The following is a ModusToolbox™ code snippet for measuring thermistor temperature using the PSOC™ 4100S Plus or PSOC™ 4100S Max device. The implementation uses a 10 kΩ reference resistor and temperature results are printed via UART. This version does not utilize an opamp.

```
#define ADCMax 4096  // Maximum counts from an 12 bit Resolution ADC
#define Vref 5       // Fullscale Voltage = 5 V
#define Beta 3435    // Beta Values in ncp18xh103f03rb Datasheet
#define T0 298.15    // Nominal Temperature Usually 25 deg celsius = 298.15 k
#define R0 10000     // Resistance at Nominal Temperature

int main(void)
{

    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init();
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Initialize the SAR ADC with the device configurator generated structure*/
    result = Cy_SAR_Init(SAR0, &pass_0_sar_0_config);
    if (result != CY_SAR_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Initialize to Debug UART port */
    Initialize_UART();
    /* Enable the SAR ADC */
    Cy_SAR_Enable(SAR0);

    /* Print message */
    Cy_SCB_UART_PutString(UART_HW,"\x1b[2J\x1b[;H");
    Cy_SCB_UART_PutString(UART_HW,"-----------------------------------------------------\r\n");
    Cy_SCB_UART_PutString(UART_HW,"PSOC 4 MCU: SAR ADC NTC TEMPERATURE MEASUREMENT\r\n");
    Cy_SCB_UART_PutString(UART_HW,"-----------------------------------------------------\r\n\n");
    for (;;)
    {
        /* Start the continuous conversion */
        Cy_SAR_StartConvert(SAR0, CY_SAR_START_CONVERT_SINGLE_SHOT);
        /* Wait till the sample is ready */
        Cy_SAR_IsEndConversion(SAR0, CY_SAR_WAIT_FOR_RESULT);
        /* Get the result from Input 0 */
        adcResult0 = Cy_SAR_GetResult16(SAR0, 0);
        /* Convert the result from Input 0 to millivolts */
        Thermistor_Voltage = Cy_SAR_CountsTo_mVolts(SAR0, 0, adcResult0) ;

        /*****************************************************************************
         *      1.Thermistor Circuit is a Voltage Divider Circuit(See Schematics)
```

```
            *      2.Voltage across Thermistor is measured from SAR ADC -> Thermistor_Voltage);
            *      3.Find the NTC sensor resistor from the Voltage Divider equation.
            *      4.Vt = (Rt/(Rt+10k))*Vref
            *      5.Rt = (Vt*10k)/(Vref-Vt) Where Vt in in mv, and Vref=5 V.
            *      6.Rt is the resistance of NTC sensor
            ********************************************************************************/

             thermistorResistance_In_ohm =  ((Thermistor_Voltage*10)/(Vref -
(Thermistor_Voltage/1000)));

             /* Convert Thermistor Resistance in Kilo ohm so that it is easy to view when
displayed in terminal*/
            thermistorResistance_In_Kohm = thermistorResistance_In_ohm/1000;
            /*******************************************************************************
            *      1.Caluclate Temperature of Thermistor using Beta parameter equation
            *      2.1/Tt = 1/T0 + ((1/Beta) * ln(Rt/R0))
            *      3.Tt = (T0*Beta)/(Beta + T0*ln(Rt/R0))
            *      4.Thermistor temperature in Kelvin
            ********************************************************************************/
            temperature_in_Kelvin = (T0*Beta)/(Beta+(T0*log(thermistorResistance_In_ohm/R0)));
            /* Convert Kelvin to celsius */
            temperature_in_Celsius = temperature_in_Kelvin - 273.15;

            Print_ADC(0,adcResult0);
            Cy_SysLib_Delay(1000);
        }
    }

    static void Initialize_UART(void)
    {
        Cy_SCB_UART_Init(UART_HW , &UART_config, &UART_context);
        Cy_SCB_UART_Enable(UART_HW);
    }
    static void Print_ADC(uint8_t adc_ch, int adc_value)
    {
        /* Variables to store Integer and Fraction parts of Thermistor Resistance */
        uint32_t  intPart_of_thermistorResistance,fracPart_of_thermistorResistance;
        /* Variables to store Integer and Fraction parts of Thermistor Temperature */
        uint32_t  intPart_of_temperature_in_Celsius,fracPart_of_temperature_in_Celsius;
        /* Declaring Buffers to store results */
        char ADC_Count_buff[100];
        char Thermistor_Voltage_buff[100];
        char Thermistor_Resistance_buff[100];
        char Thermistor_Temperature_buff[100];
        switch(adc_ch)
        {
            /* Channel 0 */
            case 0:
                /* Integer part of Kilo ohm Thermistor Resistance */
                intPart_of_thermistorResistance = (int)thermistorResistance_In_Kohm;
                /* Fraction  part of Kilo ohm Thermistor Resistance */
                fracPart_of_thermistorResistance = (int)((thermistorResistance_In_Kohm -
intPart_of_thermistorResistance) * 1000);
```

```
            /* Integer part of Thermistor Temperature in Celsius */
            intPart_of_temperature_in_Celsius = (int)temperature_in_Celsius;
            /* Fraction part of Thermistor Temperature in Celsius */
            fracPart_of_temperature_in_Celsius = (int)((temperature_in_Celsius -
intPart_of_temperature_in_Celsius) * 1000);
            /*Update buffers for Input 0 */
            sprintf(ADC_Count_buff,"ADC Count is %d\r\n",adc_value);
            sprintf(Thermistor_Voltage_buff,"Thermistor Voltage at Channel 0= %d
mV\r\n",Thermistor_Voltage);
            sprintf(Thermistor_Resistance_buff, "Thermister Resistance= %d.%03d kohm\r\n",
intPart_of_thermistorResistance, abs(fracPart_of_thermistorResistance));
            sprintf(Thermistor_Temperature_buff, "Temperature = %d.%03d C\r\n",
intPart_of_temperature_in_Celsius, abs(fracPart_of_temperature_in_Celsius));
            break;
        /* Channel 1 */
        case 1:
            /*Update buffer for Input 1 result*/
            sprintf(ADC_Count_buff,"ADC Result Channel 1= %d mV\r\n",adc_value);
            break;
        /* Channel 2 */
        case 2:
            /*Update buffer for Input 2 result*/
            sprintf(ADC_Count_buff,"ADC Result Channel 2= %d mV\r\n",adc_value);
            break;
        /* Channel 3 */
        case 3:
            /*Update buffer for Input 2 result*/
            sprintf(ADC_Count_buff,"Die Temperature= %d degC\r\n\n",adc_value);
            break;
    }
    /*Print Updated buffer on UART port*/
    Cy_SCB_UART_PutString(UART_HW,"-----------------------------------------------------\r\n");
    Cy_SCB_UART_PutString(UART_HW,ADC_Count_buff);
    Cy_SCB_UART_PutString(UART_HW,Thermistor_Voltage_buff);
    Cy_SCB_UART_PutString(UART_HW,Thermistor_Resistance_buff);
    Cy_SCB_UART_PutString(UART_HW,Thermistor_Temperature_buff);
    Cy_SCB_UART_PutString(UART_HW,"-----------------------------------------------------\r\n");
}
```

The above code is applicable for both PSOC™ 4100S Plus and PSOC™ 4100S Max devices, the only difference is to change the BSP for both the devices in the ModusToolbox™ IDE.

## 4.3 Accuracy analysis

Resistive sensor measurement accuracy depends on the accuracy of the resistance measurements. When measuring resistances, the following are sources of error:

- Tolerance of reference resistor
- ADC offset and gain
- ADC non-linearity

See the error in physical quantity, for example, temperature or pressure section to calculate the error in the physical quantity based on the error in the resistance measurement.

### 4.3.1 Tolerance of reference resistor

Start by adding an error $\Delta R_{ref}$ to $R_{ref}$ in (2). This results in an error $\Delta R_{sensor}$ added to $R_{sensor}$.

$$R_{sensor} + \Delta R_{sensor} = \frac{(V_{SENSOR} - V_{GND})}{(V_{SOURCE} - V_{SENSOR})} * \left(R_{ref} + \Delta R_{ref}\right) \quad (3)$$

The change in $(V_{SENSOR}–V_{GND})/(V_{SOURCE}–V_{SENSOR})$ is small and is ignored to simplify the calculation. You get (6) by dividing (3) by (2).

$$\frac{R_{sensor} + \Delta R_{sensor}}{R_{sensor}} \cong \frac{R_{ref} + \Delta R_{ref}}{R_{ref}} \quad (4)$$

$$1 + \frac{\Delta R_{sensor}}{R_{sensor}} \cong 1 + \frac{\Delta R_{ref}}{R_{ref}} \quad (5)$$

$$\frac{\Delta R_{sensor}}{R_{sensor}} \cong \frac{\Delta R_{ref}}{R_{ref}} \quad (6)$$

As seen in the above equation, the % error introduced in $R_{sensor}$ is approximately the same as the % error in $R_{ref}$. The % error in $R_{ref}$ is the tolerance of the resistor. For example, if the tolerance of $R_{ref}$ is 0.1%, the maximum error introduced by $R_{ref}$ in the resistance measurement is approximately 0.1%.

Table 1 shows the actual errors in measured resistance for a 10 kΩ reference resistance. The % error in the measured value of $R_{sensor}$ is the same for all values of $R_{sensor}$. A 0.1% error in $R_{ref}$ introduces a 0.1% error in $R_{sensor}$. A 5% error in $R_{ref}$ introduces a 4.76% error in $R_{sensor}$.

**Table 1 Error in resistance measurement for various values of error in reference resistance**

| $R_{sensor}$ | Error in $R_{ref}$ | | | |
|---|---|---|---|---|
| | **0.10%** | **1.00%** | **2%** | **5%** |
| 10 kΩ | 0.10% | 0.99% | 1.96% | 4.76% |

### 4.3.2 ADC offset and gain errors

Introduce offset error Vos and gain error x to the (2).

$$R_{sensor} = \frac{x(V_{SENSOR} + V_{os}) - x(V_{GND} + V_{os})}{x(V_{SOURCE} + V_{os}) - x(V_{SENSOR} + V_{os})} * R_{ref} \quad (7)$$

$$R_{sensor} = \frac{\cancel{x}\left(V_{SENSOR} + \cancel{V_{os}} - V_{GND} - \cancel{V_{os}}\right)}{\cancel{x}\left(V_{SOURCE} + \cancel{V_{os}} - V_{SENSOR} - \cancel{V_{os}}\right)} * R_{ref} \quad (8)$$

(8) shows that both the offset and gain errors are canceled in the potential divider method. Therefore, ADC offset and gain errors do not affect the accuracy of measurement.

When using differential measurement, the offset voltage of the ADC can be measured by adding one more differential channel and shorting both terminals to the ground. Later the offset count can then be subtracted from the counts that correspond to other channels to remove the offset error.

### 4.3.3 ADC nonlinearity error

The SAR ADC in PSOC™ 4100S Plus device has a maximum INL of ±1.7 LSB. For a 1.2 V range and 12-bit resolution, the INL error is approximately 0.5 mV.

Similarly, SAR ADC in PSOC™ 4100S Max device has a maximum INL of ±3 LSB. For a 1.2 V range and 12-bit resolution, the INL error is approximately 0.9 mV.

The worst-case error occurs when the INL voltage affects both the numerator and denominator of (2). Consider the following example, where:

$R_{sensor}$ = 7 kΩ $R_{ref}$ = 10 kΩ Excitation voltage = 1.2 V

The worst-case error in the measured resistance due to the INL error is 10 Ω, which is 0.14% error. The worst-case error in the measured resistance due to ADC INL error for various values of $R_{sensor}$ is shown in the following figure.

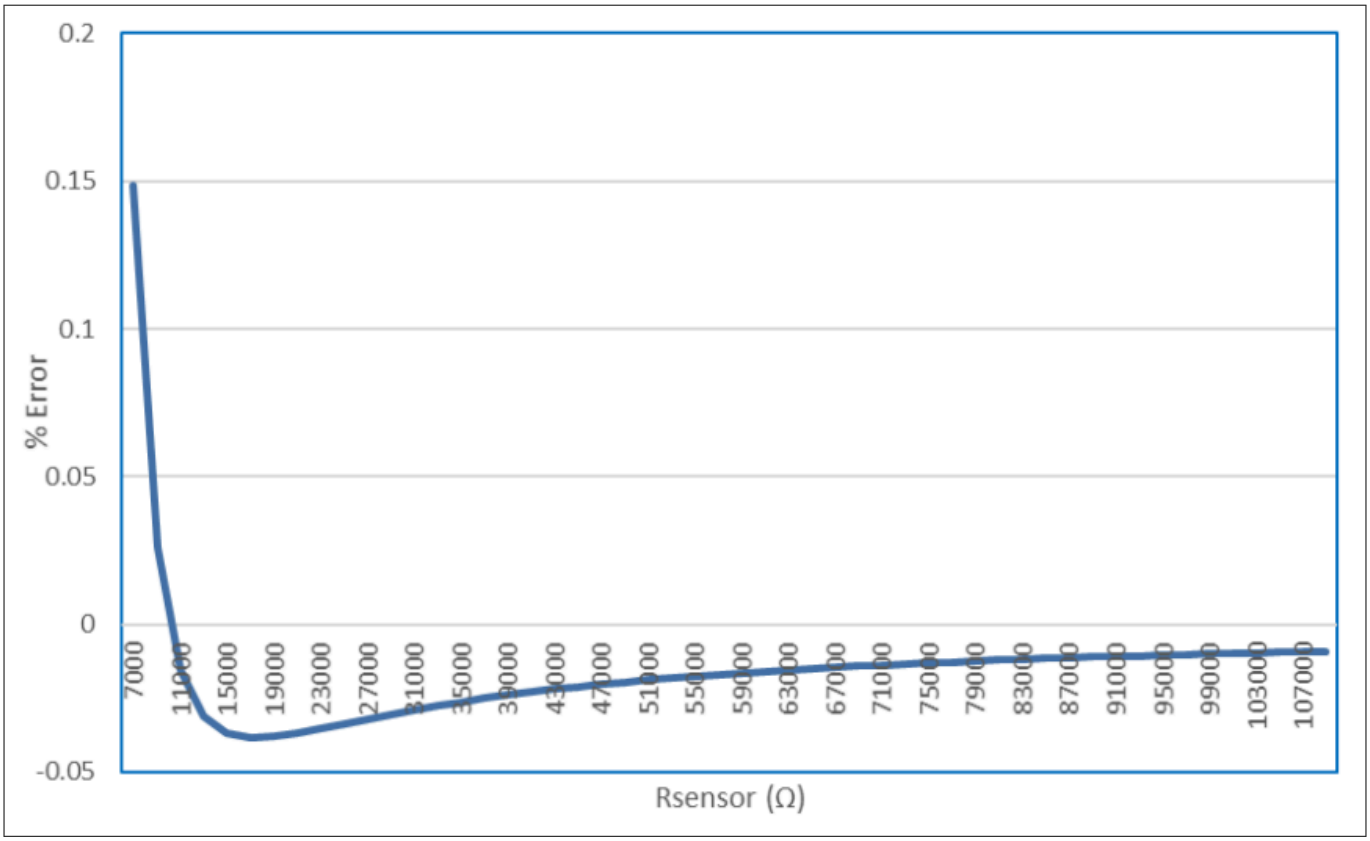



**Figure 8 Worst case error in resistance due to INL versus $R_{sensor}$**

### 4.3.4 Error in physical quantity (temperature or pressure)

This section describes the steps to calculate the error in the measured physical quantity, for example ΔT in case of temperature, due to the error in the measured resistance value ($\Delta R_{sensor}$).

1. Calculate the error in sensor resistance ($\Delta R_{sensor}$) for a known set of $R_{ref}$, $\Delta R_{ref}$, and $R_{sensor}$ from (6)
2. Calculate the absolute value of the deviated sensor resistance $R'_{sensor}$ from (9)

$$R'_{sensor} = R_{sensor} \pm \Delta R_{sensor} \quad (9)$$

3. From the sensor manufacturer's datasheet, find the physical quantity corresponding to $R_{sensor}$ and $R'_{sensor}$
4. Take the difference between the physical quantity (like temperature or pressure) values calculated in Step 3 to calculate the error in physical quantity due to the variation in $R_{ref}$. This value corresponds to the error in a physical quantity

#### 4.3.4.1 Reference resistor selection

To achieve the maximum resolution at either extreme of temperature, the reference resistance should be close to the resistance of the thermistor at the middle of its temperature range. If you are more interested in measuring the temperature at one extreme, then the reference resistance should match the resistance of the thermistor at the extreme temperature being measured.

#### 4.3.4.2 Conversion accuracy

The accuracy of converting resistance to physical quantity depends on the accuracy of curve fitting to match the nonlinear characteristics of the sensor. For example, in the case of negative temperature coefficient (NTC) thermistors, the resistance of the thermistor decreases as the temperature rises. The variation of resistance with temperature is nonlinear. In the thermistor calculator component, the conversion error across a temperature range of –40°C to 125°C is less than 0.01°C.

# 5 AFE for capacitive sensor

In capacitive sensors, the capacitance between the sensor terminals varies with respect to the physical quantity being measured. A few examples of capacitive sensors are:

- Humidity sensors
- Pressure sensors
- Proximity sensors

This section describes two AFE designs for measuring a capacitive sensor. It also explains the design implementation using PSOC™ 4100S Plus and PSOC™ 4100S Max devices. An Accuracy analysis section explains the errors associated with one of the capacitance measurement methods and recommendations to handle such errors.

## 5.1 Single slope method

In this method, a constant current source is used to charge the capacitor. The time taken to charge the capacitor to a known voltage can be used to measure capacitance.

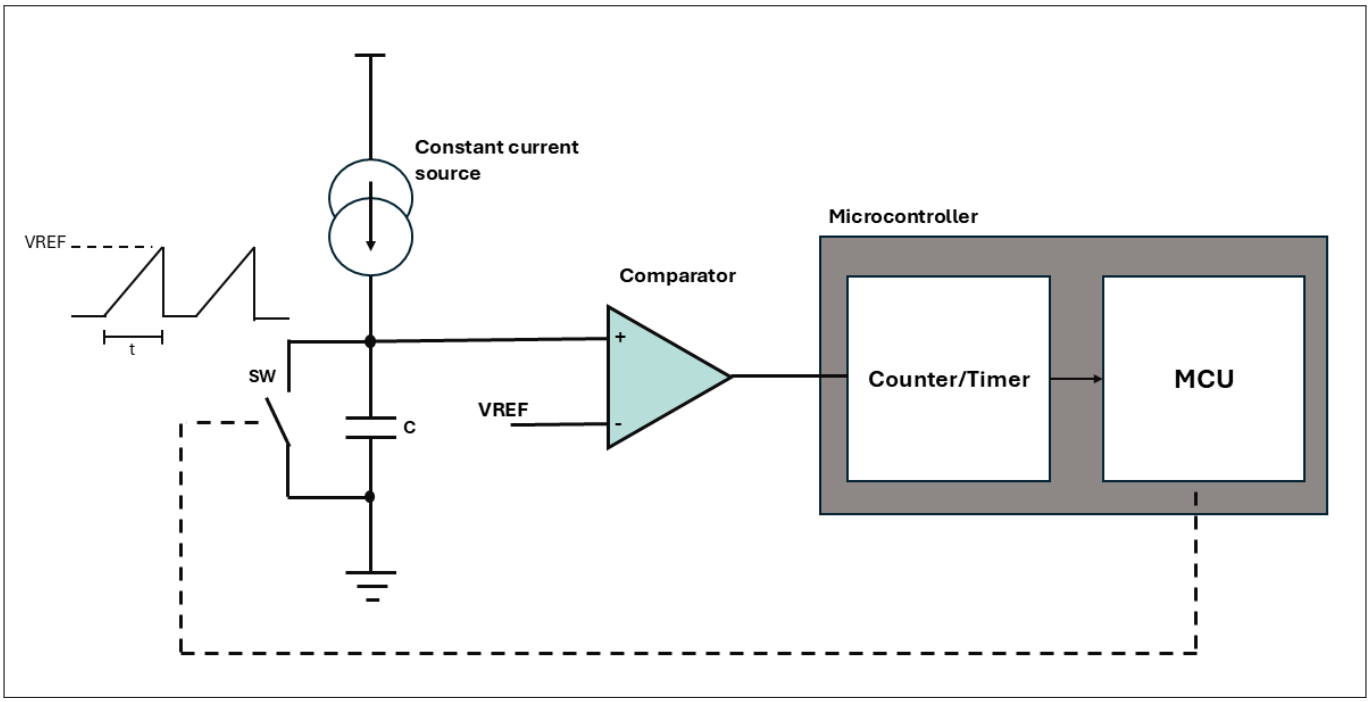



**Figure 9 Single slope method to measure capacitance**

In the above circuit, the microcontroller keeps the switch (SW, could be a GPIO or an external FET) closed, so holding the capacitor in a discharged state. Then the microcontroller releases the switch. The current source charges the capacitor and the voltage across the capacitor ramps up linearly. When the voltage across the capacitor becomes greater than VREF, the comparator output goes HIGH. The microcontroller uses the timer/counter to measure this time duration and then calculates the capacitance (C) using (10).

$$C = \frac{I * t}{V} \tag{10}$$

Where,

I = Current from the constant current source

V = Reference voltage of the comparator

t = Time taken to charge the capacitor to VREF

The accuracy of this method depends on the accuracy of the constant current source that is, the comparator reference voltage and the timer.

## 5.2 Sigma delta modulator method

In this method, a sigma delta modulator converts the capacitance into a digital bit stream, where the density of the bit stream is directly proportional to the capacitance value. Figure 10 shows the sigma delta modulator.

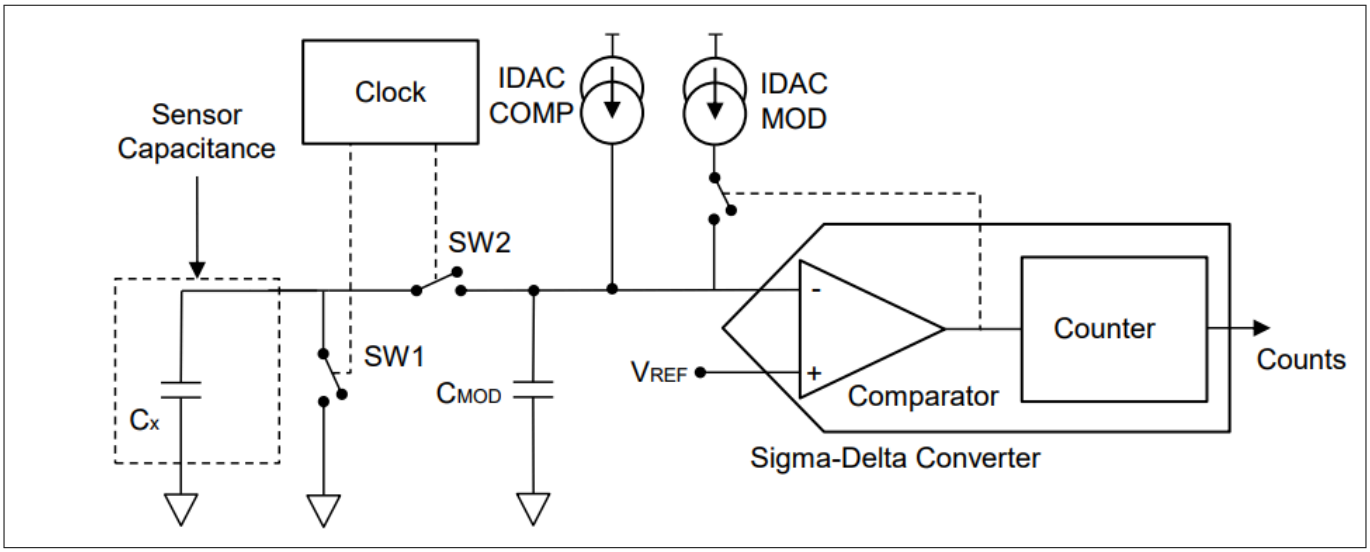



**Figure 10 Sigma delta modulator**

IDACMOD is a current source that charges a modulation capacitor CMOD. When the voltage across CMOD goes above VREF, the output of the comparator goes low and turns off IDACMOD.

Cx is the sensor capacitance to be measured. SW1 and SW2 along with Cx form a switch capacitor cell. SW1 and SW2 are driven out of phase by a clock source. When SW2 is ON, charge is transferred from CMOD to Cx. When SW1 is ON, Cx is discharged. So Cx, SW1, and SW2 together act as a resistor discharging CMOD. As CMOD is discharged, the voltage drops. When the voltage drops below VREF, the output of the comparator goes HIGH and turns on IDACMOD, which restarts the charging cycle. IDACCOMP is a fixed current source that can be used with IDACMOD to increase the dynamic range of measurement.

Figure 11 shows the output of the comparator and the voltage across CMOD.

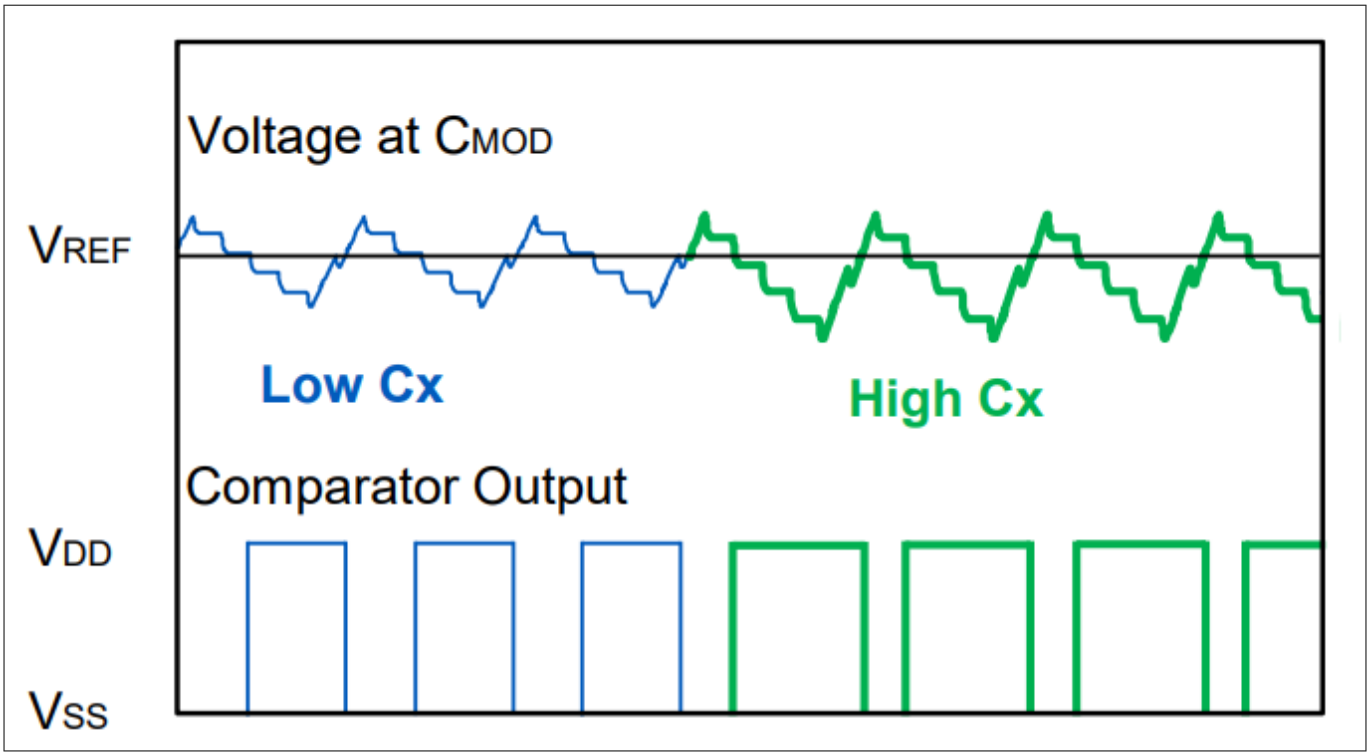



**Figure 11 Output of comparator for different $C_X$ values**

A smaller Cx (signals shown in blue) results in less charge taken out of CMOD, so letting CMOD charge faster to VREF. This results in the comparator output staying HIGH for a shorter period of time. A larger Cx (signals shown in green) results in more charge taken out of CMOD, so increasing the time to charge CMOD. This results in comparator output staying HIGH for a longer period of time. The counter integrates the output of the comparator over a given period of time and the output of the counter is directly proportional to the capacitance.

The following equation shows the relationship between the output of the counter, also called “raw count”, and the capacitance.

$$rawcount = \left(2^N - 1\right)\frac{(V_{DD} - V_{REF})F_{SW}}{I_{MOD}}C_S - \left(2^N - 1\right)\frac{I_{COMP}}{I_{MOD}} \quad (11)$$

Where,

$C_S$ = Capacitance being measured

N = Resolution of the counter in bits

$F_{SW}$ = Frequency at which SW1 and SW2 are operated

$I_{MOD}$ = Current from $IDAC_{MOD}$

$I_{COMP}$ = Current from $IDAC_{COMP}$

$V_{REF}$ = Reference voltage of comparator

As can be seen from the above equation, the accuracy of measurement depends on $IDAC_{MOD}$, $IDAC_{COMP}$, $V_{DD}$, and $V_{REF}$. An error in any of these parameters introduces an error in measured capacitance.

To compensate the error in these parameters, a single fixed capacitor ($C_{fixed}$) with good tolerance and temperature stability is used. Both the sensor capacitor and the fixed capacitor are measured with the sigma delta modulator and the value of the sensor capacitor can be found from the ratio of raw counts.

$$C_{sensor} = \frac{C_{fixed}}{RawCounts_{C_{fixed}}} * RawCounts_{C_{sensor}} \quad (12)$$

Though this method is complex than the single slope methods, all the hardware necessary to implement this method is located inside PSOC™ 4100S Plus in the capacitive sigma delta (CSD) block. This makes it easy to use this method.

## 5.3 Measurement using multi-sense converter (MSC) peripheral

In this method, the MSC block converts the capacitance into a digital bit stream (called as "Rawcounts"), where the density of the bit stream is directly proportional to the capacitance value. Figure 12 shows the multi-sense converter (MSC) located in the PSOC™ 4100S Max device.

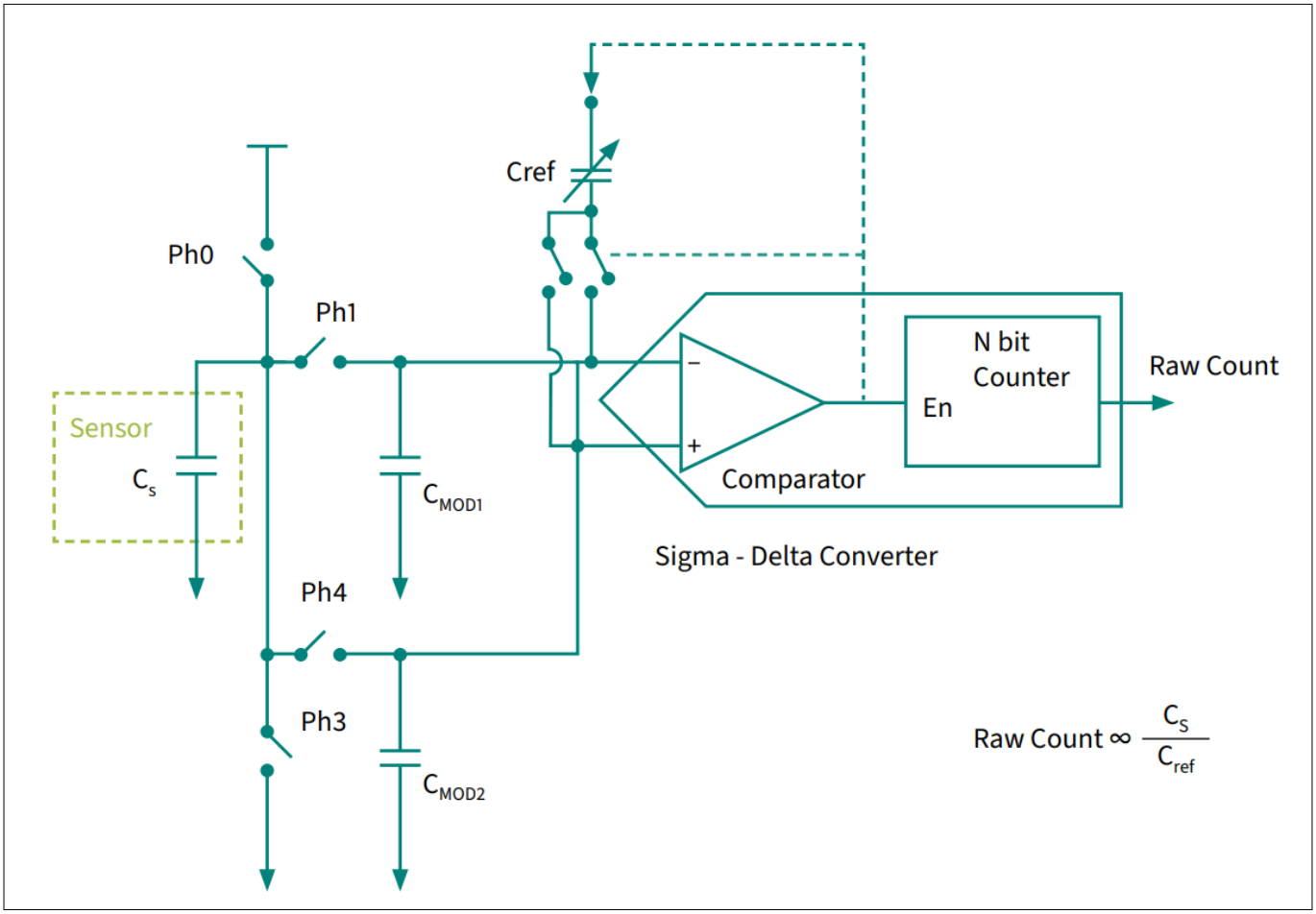



**Figure 12 Multi-sense converter (MSC)**

Figure 12 shows a basic representation of capacitive sensing using the MSC block. The sensor, referred to as Cs, alternates its connection between CMOD1, CMOD2, ground, or power through switches Ph0 to Ph3. When the sensor is connected to the CMOD capacitors, a charge transfer occurs between the sensor and the CMOD capacitors.

A comparator is used to detect changes in the charge on the CMOD capacitors. Once a change is detected, it activates the counter and clock, which simultaneously drive the cap digital-to-analog converters(DACs). The cap DACs work to replace the charge on the CMOD capacitors. When the comparator determines that all the charge has been replenished, the DACs are disabled. At the same time, the counter stops, providing a count value that is directly proportional to the capacitance of the sensor, as described in the following equation:

$$Rawcount \propto \frac{C_S}{C_{REF}} \tag{13}$$

Figure 13 shows the output waveform of comparator and voltage difference between $C_{MOD1}$ and $C_{MOD2}$ capacitors along with how does the counter increment when the comparator produces HIGH.

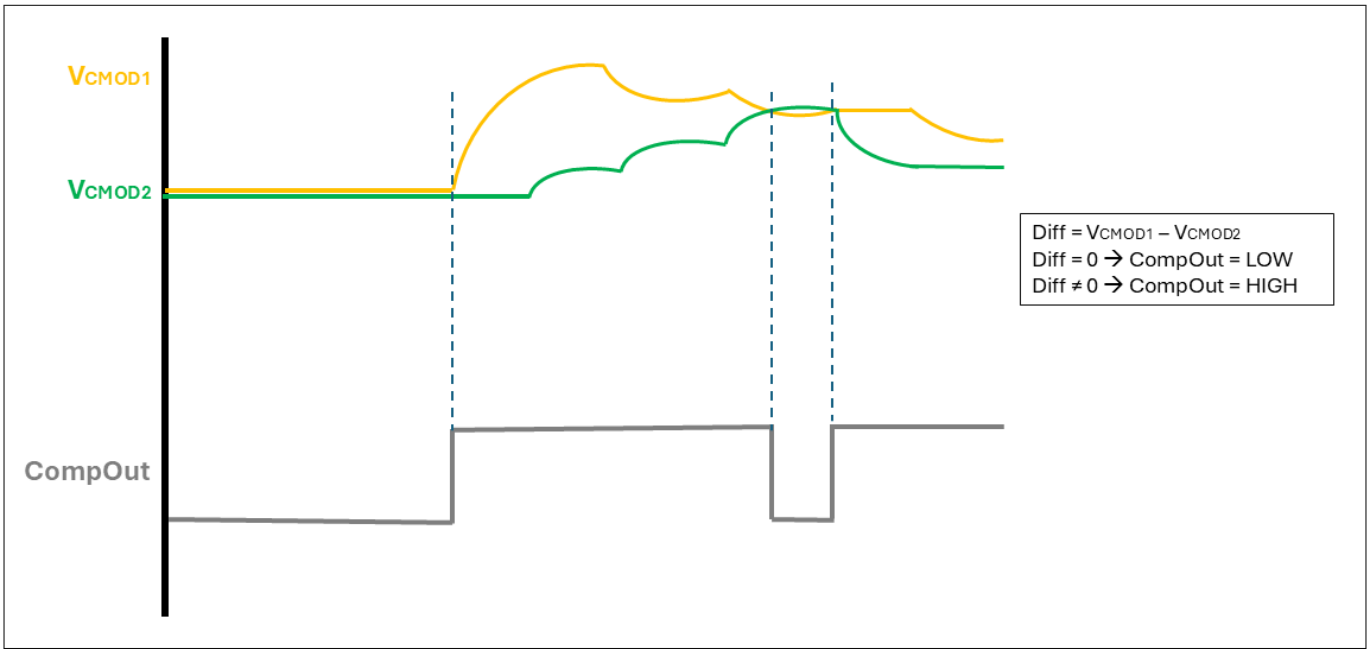



**Figure 13 Waveforms of voltage across sensor, CMOD capacitors, and comparator output**

From the diagram (Figure 13), infer that when the sensor capacitance (Cs) increases, it draws more charge from the CMOD capacitors. As a result, replenishing this charge takes additional time. During this charging period, the comparator output remains HIGH. As the system continues counting while the comparator is HIGH, the counter value representing the raw capacitance is directly proportional to the sensor's capacitance.

The following equation shows the relationship between the output of the counter, also called “Rawcount”, and the capacitance.

$$Rawcount = \frac{N_{SUB}}{C_{REF}}\left(C_S - 2\frac{F_{COMP}}{F_{SW}}C_{COMP}\right) \tag{14}$$

Where,

$N_{SUB}$ = Number of sub-conversions

$C_{REF}$ = Reference capacitance ($CDAC_{MOD}$)

$C_S$ = Sensor capacitance

$F_{COMP}$ = Compensator clock frequency

$F_{SW}$ = Sense clock frequency

$C_{COMP}$ = Compensator capacitance ($CDAC_{COMP}$)

As can be seen from the above equation, the accuracy of measurement depends on $CDAC_{MOD}$, $CDAC_{COMP}$, $V_{DD}$, and $V_{REF}$. An error in any of these parameters introduces an error in measured capacitance.

To compensate the error in these parameters, a single fixed capacitor ($C_{fixed}$) with good tolerance and temperature stability is used. Both the sensor capacitor and the fixed capacitor are measured with the MSC peripheral and the value of the sensor capacitor can be found from the ratio of raw counts.

$$C_{sensor} = \frac{C_{fixed}}{RawCounts_{C_{fixed}}} * RawCounts_{C_{sensor}} \quad (15)$$

Though this method is complex than the single slope methods, all the hardware necessary to implement this method is located in PSOC™ 4100S Max's multi-sense converter (MSC) block. This makes it easy to use this method.

## 5.4 Impact of parasitic capacitance

The parasitic capacitance of the PCB trace connecting the sensor capacitance to the sensing hardware and pin capacitance are added to the result, introducing an offset error.

There are two ways to address this offset caused by PCB trace. Keep the PCB trace as short as possible to reduce the parasitic capacitance. This method is used for the fixed capacitor ($C_{fixed}$), which is usually not practical for the sensor capacitor. In such cases, another method will be used, where the parasitic capacitance of the system is measured by taking the initial measurements with the sensor disconnected from PSOC™ 4100S Plus or PSOC™ 4100S Max device. The result can then be subtracted from the sensor capacitor measurements.

## 5.5 Design implementation for capacitance measurement

Figure 14 shows that how does the CSD peripheral/MSC peripheral can be used to measure the humidity sensor capacitance.

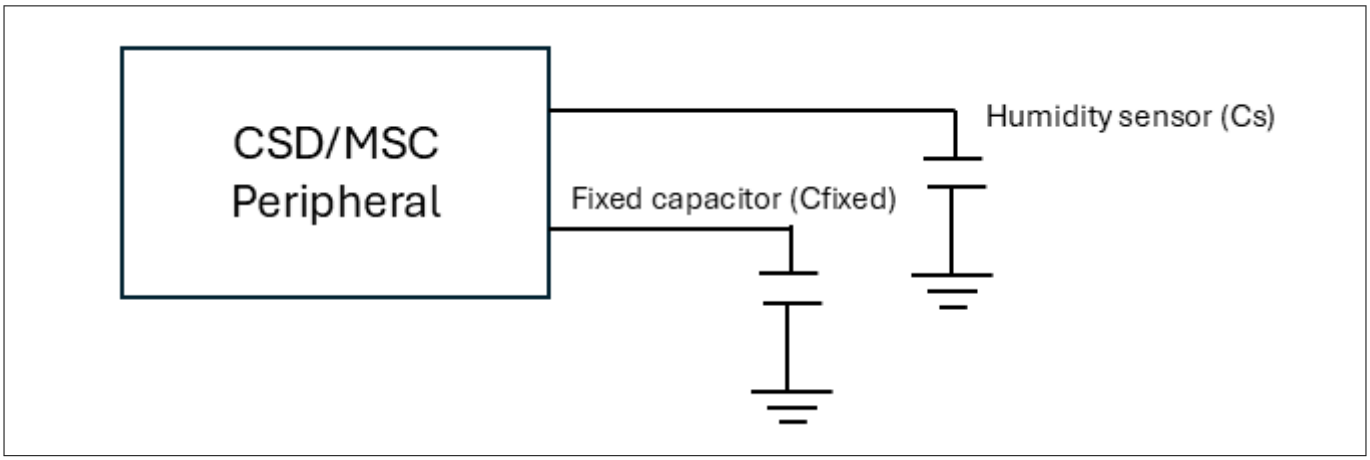



**Figure 14 Capacitance measurement block diagram**

Using the CAPSENSE™ peripheral (CSD/MSC) PSOC™ 4100S Plus/PSOC™ 4100S Max measures the raw counts ($RawCount_{CS}$) that correspond to the fixed capacitor value ($C_{fixed}$). Then the raw counts with the sensor disconnected is measured. These raw counts ($RawCount_{COS}$) correspond to the parasitic capacitance ($C_{os}$). As the fixed capacitor ($C_{fixed}$) is known, the value of the parasitic capacitance can be found using (12). The raw counts with the sensor connected ($RawCounts_{CS}$) is measured. The value of the sensor capacitor is calculated using the following equation.

$$C_S = \frac{C_{fixed} * \left(RawCount_{C_S} - RawCount_{C_{OS}}\right)}{RawCount_{C_{fixed}} - RawCount_{C_{OS}}} \quad (16)$$

Where,

$C_{fixed}$ = Fixed capacitance value

$C_s$ = Sensor capacitance value

$RawCount_{Cs}$ = Raw count from measuring humidity sensor capacitance
$RawCount_{Cos}$ = Raw count value that corresponds to Cos (caused by PCB trace and GPIO capacitance)
$RawCount_{Cfixed}$ = Raw count from measuring reference capacitance
Once sensor capacitance is calculated, humidity is calculated by using the following equation:

$$\%RH = \frac{C_S - C_{NOM}}{Sensitivity} + \%RH_{NOM} \quad (17)$$

Where,
%RH = Relative humidity
$C_s$ = Measured capacitance of humidity sensor
$C_{NOM}$ = Nominal capacitance of sensor (for the HS1100, this is 180 pF)
$\%RH_{NOM}$ = Nominal humidity of the sensor (for the HS1100, this is 55%RH)
Sensitivity = pF/%RH value of the sensor (for the HS1100, this is 0.34 pF/%RH)

## 5.6 Accuracy analysis

The system accuracy of humidity measurement depends on the accuracy of capacitance measurement by CAPSENSE™ Component and the sensor linearity.
As seen in the previous section, capacitance is calculated by using (18).

$$C_S = \frac{C_{fixed} * \left(RawCount_{C_S} - RawCount_{C_{OS}}\right)}{RawCount_{C_{fixed}} - RawCount_{C_{OS}}} \quad (18)$$

From (18), the accuracy of capacitance measurement depends on the accuracy of fixed capacitor ($C_{fixed}$). The error in the measured capacitance (C) is due to the change in $C_{fixed}$ is provided by (19).

$$Error = \Delta C = \frac{\pm C * \Delta C_{fixed}}{C_{fixed} \pm \Delta C_{fixed}} \quad (19)$$

# 6 Summary

This application note explains the AFE implementation for voltage, resistive, and capacitive sensors. It provides the implementation details of each element in the AFE using ModusToolbox™ IDE.

# References

### Application notes

**[1]** Infineon Technologies AG: *AN79953 – Getting started with PSOC™ 4 MCU*; Available online
**[2]** Infineon Technologies AG: *AN86233 – PSOC™ 4 MCU low-power modes and power reduction techniques*; Available online
**[3]** Infineon Technologies AG: *AN88619 – PSOC™ 4 MCU hardware design considerations*; Available online
**[4]** Infineon Technologies AG: *AN73854 – PSOC™ Creator - Introduction to bootloaders*; Available online
**[5]** Infineon Technologies AG: *AN89056 – PSOC™ 4 – IEC 60730 Class B and IEC 61508 SIL Safety Software Library*; Available online
**[6]** Infineon Technologies AG: *AN60590 – PSOC™ 3, PSOC™ 4, and PSOC™ 5LP - Temperature measurement with a diode*; Available online
**[7]** Infineon Technologies AG: *AN70698 – PSOC™ 3, PSOC™ 4, and PSOC™ 5LP - Temperature measurement with an RTD*; Available online
**[8]** Infineon Technologies AG: *AN66477 – PSOC™ and PSOC™ 5LP - Temperature measurement with a thermistor*; Available online
**[9]** Infineon Technologies AG: *AN89610 – PSOC™ Arm® Cortex® code optimization*; Available online
**[10]** Infineon Technologies AG: *AN90799 – PSOC™ 4 interrupts*; Available online
**[11]** Infineon Technologies AG: *AN86439 – PSOC™ 4 MCU - Using GPIO pins*; Available online
**[12]** Infineon Technologies AG: *AN85951 – PSOC™ 4 CAPSENSE™ design guide*; Available online
**[13]** Infineon Technologies AG: *AN92239 – Proximity sensing with CAPSENSE™*; Available online
**[14]** Infineon Technologies AG: *AN86526 – PSOC™ 4 I2C bootloader*; Available online
**[15]** Infineon Technologies AG: *AN68272 – PSOC™ 3, PSOC™ 4, and PSOC™ 5LP UART bootloader*; Available online
**[16]** Infineon Technologies AG: *AN87391 – PSOC™ 4 segment LCD drive*; Available online
**[17]** Infineon Technologies AG: *AN84858 – PSOC™ 4 programming using an external microcontroller (HSSP)*; Available online

### Datasheets

**[18]** Infineon Technologies AG: *PSOC™ 4100S Plus datasheet*; Available online
**[19]** Infineon Technologies AG: *PSOC™ 4100S Max datasheet*; Available online

### Kits

**[20]** Infineon Technologies AG: *PSOC™ 4100S Plus Prototyping Kit*; Available online
**[21]** Infineon Technologies AG: *PSOC™ 4100S Max Pioneer Kit*; Available online

### Tool

**[22]** Infineon Technologies AG: *ModusToolbox™ software*; Available online

# Glossary

**AFE**
analog front ends

**BGR**
bandgap reference voltage

**BOM**
bill of materials

**BSP**
board support package

**CSD**
capacitive sigma delta

**GPIO**
general purpose input/output

**INL**
integral nonlinearity

**LDR**
light dependent resistor

**LSB**
least significant byte

**MSC**
multi-sense converter

**NTC**
negative temperature coefficient

**PGA**
programmable gain amplifier

**RH**
relative humidity

**RTD**
resistance temperature detector

**SAR**
successive approximation register

**SCB**
Serial Communication Block

**TIA**
transimpedance amplifier

# Revision history

| Document revision | Date | Description of changes |
|---|---|---|
| ** | 2025-07-16 | Initial release |

# Trademarks

PSOC™, formerly known as PSoC™, is a trademark of Infineon Technologies. Any references to PSoC™ in this document or others shall be deemed to refer to PSOC™.

**Trademarks**

All referenced product or service names and trademarks are the property of their respective owners.

**Edition 2025-07-16**

**Published by**

**Infineon Technologies AG**

**81726 Munich, Germany**

**© 2025 Infineon Technologies AG**

**All Rights Reserved.**

**Do you have a question about any aspect of this document?**

**Email: erratum@infineon.com**

**Document reference**
**IFX-mej1744295449200**

**Important notice**

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

**Warnings**

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.