INNOVATION WITH PURPOSE
ALLEGRO microsystems

CALIBRATION OF A17700 PRESSURE SENSOR INTERFACE

By Christophe Lutz and Cédric Gillet
Allegro MicroSystems

INTRODUCTION

Many applications require accurate pressure sensing and are exposed to large temperature variations, e.g., gas or liquid monitoring. Generally, the pressure is measured by means of a strain gauge, which is constituted by piezoresistive elements built onto a deflecting membrane. The sensing elements have changing responses with pressure, but unfortunately also drift with temperature. A pressure sensor interface is typically implemented to remove the unwanted temperature dependence and improve significantly the pressure accuracy.

Allegro's A17700 pressure sensor signal conditioner is qualified over a temperature range from –40°C to 150°C and is featured with a Polynomial Compensation up to a 4th order, Poly(4,4), to be associated with any type of Wheatstone bridge. Resulting output error post compensation can typically be maintained within 0.35%.

This application note aims at providing a complete example of calibration using A17700 Pressure Sensor Interface.

Strain Gauge Characteristic

The pressure sensing system is constituted of a membrane, and one to four piezoresistors in a Wheatstone bridge arrangement connected to a pressure interface. A difference of pressure on the sides of membrane creates a net force on the membrane which deflects. The membrane is subjected to a distribution of compressive and tensile stress; this is also true at the locations of the piezoresistors which now exhibit different resistances. As a result, a net differential voltage is created at the output of the bridge.

This structure is particularly vulnerable to temperature for several reasons. First, the membrane mechanical properties are altered with temperature. Secondly, the conductance of each resistor drifts due to microscopic fluctuations. Lastly, both are affected by thermal dilation which in turn affects the stress profiles at the interface between the piezoresistor and the membrane. The following figure displays typical bridge characteristics when passed through an untrimmed A17700 interface, which is at this stage perfectly equivalent to a linear differential operational amplifier.

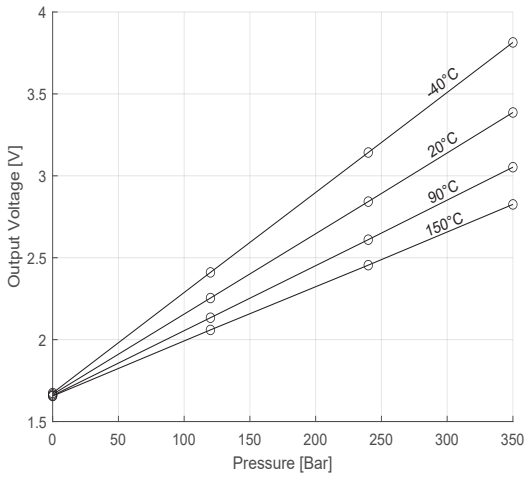

Figure 1: Piezoresistive Bridge output through A17700 Sensor Interface before coefficient programming

The A17700's role is to remove the temperature dependence of the bridge in order to have an accurate and unbiased measurement of the pressure applied to the membrane. For that sake, the A17700 embeds an internal temperature sensor and a polynomial correction block whose role is to establish the linearity of the output with pressure only and to set the output dynamics as desired. The following figure shows a general representation of the output of a calibrated pressure sensor, which includes the normal operating range delimited by (P_{MIN}, V_{MIN}) and (P_{MAX}, V_{MAX}), an abnormal pressure range ($P>P_{MAX}$ or $P<P_{MIN}$) for which the performance extends until a saturation is reached (determined by the programming of the interface clamps), and a range that serves for diagnostics (pull-up or pull-down voltage in case of safety flag tripping or broken output).

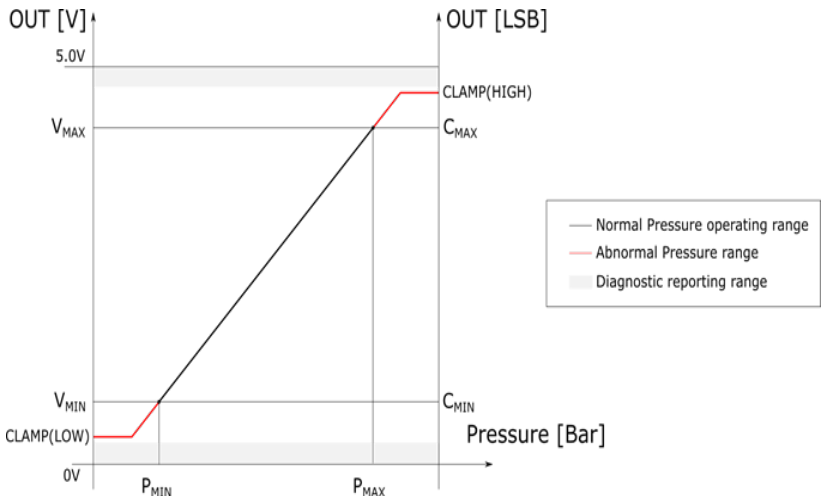

Figure 2: Typical Characteristic of a Calibrated Pressure Sensor.

Pressure Interface Integrated Circuits

The A17700 is an interface offering the following top-level features:

- 3.3 V supply for pressure sensing element (Bridge supply)
- An optimized set of front-end amplifiers to achieve best signal to noise ratio (Analog Front End)
- An optimized 16-bit ADC
- Embedded programmable filters to trade resolution and speed (20 kHz bandwidth option offers a response time below 200 µs)
- A polynomial compensation block to remove the temperature dependence of the bridge signal—Poly(4,4)
- Analog or digital output

The following figure shows the typical application circuits of the A17700 interfaces:

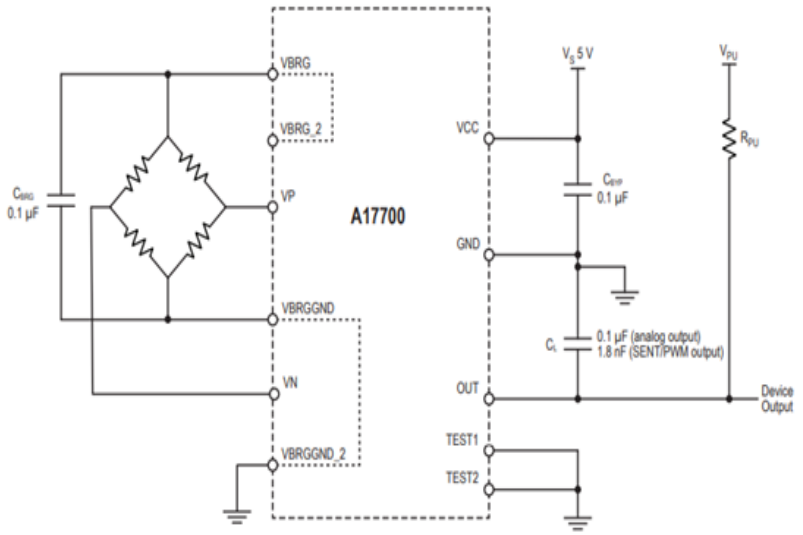

Figure 3: Typical Application Circuit of A17700 Interfaces

Interface Calibration Methodology

The calibration is performed in two steps. The role of the first step is to find the configuration of amplifiers settings that maximizes the filling of the ADC (up to some tolerances). The role of the second step is to determine the coefficients of the polynomial compensation.

Analog Front End (Room Temperature)

After the bridge is connected to the device, the first step is to adjust the polarity of the bridge. The role of the negative pin VN and positive pin VP can be interchanged through programming. It offers flexibility in routing the wires bounded to the membrane. The polarity bit must be set in order to have increasing output voltage when the pressure is increasing.

Table 1: Polarity bit EEPROM Information

Name	Address	Bit	Values	Behavior
afe_polarity	0x17	[12]	0	Default VP/VN
–	–	–	1	Interchanged VP/VN

The second step is to adjust the coarse gain, offset and fine gain of the analog front end in order to ensure a proper filling of the ADC.

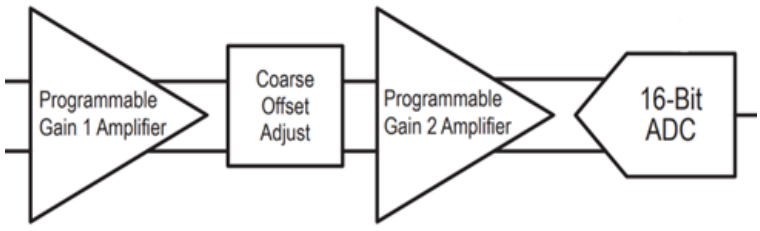

Figure 4: Schematic of the Analog Front End Chain

The afe_amp_gain1 is a coarse gain amplifier which is constant at a production scale and kept unchanged. The gain 1 is initially kept in its default setting (0b00) and increased if an upper saturation on afe_amp_gain2 is experienced downstream the analog calibration.

To adjust the afe_amp_offset and the afe_amp_gain2, it is necessary to know the largest dynamic of the bridge differential voltage. The minimum pressure and maximum pressure must be applied to readout the minimum and maximum output (analog voltage or digital word for the digital protocols), at a given temperature (generally room temperature).

Afe_amp_offset must be adjusted to have the median of the previous values match the midrange of the output signal (2.5 V or 2048 LSB). Afe_amp_gain2 must be adjusted to have the peak-to-peak amplitude represent 50% of the output peak-to-peak amplitude. It is recommended to keep headroom in the filling of the ADC to avoid saturation under all operating conditions (especially over temperature). A typical filling of 50% is advised. In this manner, the adjustment can be performed at room temperature only.

These steps require the access to the following memory registers:

Table 2: Analog EEPROM Information

Name	Address	Bits	Values	Behavior
afe_amp_gain1	0x16	[4:3] Unsigned	2'b00 2'b01 2'b10 2'b11	G1 = 3 G1 = 6 G1 = 9 G1 = 12
afe_amp_offset	0x16	[14:9] Signed	6'b000000 6'b000001 … 6'b011111 6'b100000 6'b100001 … 6'b111111	OFF = 0 V/V OFF = 8.1e–3 … OFF = 249e–3 OFF = – 257.1e–3 OFF = – 249e–3 … OFF = – 8.1e–3
afe_amp_gain2	0x16	[8:5] Unsigned	4'b0000 4'b0001 … 4'b1111	G2 = 1 G2 = 1.4 … G2 = 7

Note that analog trimming must be settled before moving to the digital trimming. In Allegro's "A17700 Samples Programmer", "AFE Trim" Tab, the front-end analog trimming is automatically performed with two points measurements (Min Pressure/Max Pressure – See Figure 5 and Figure 8).

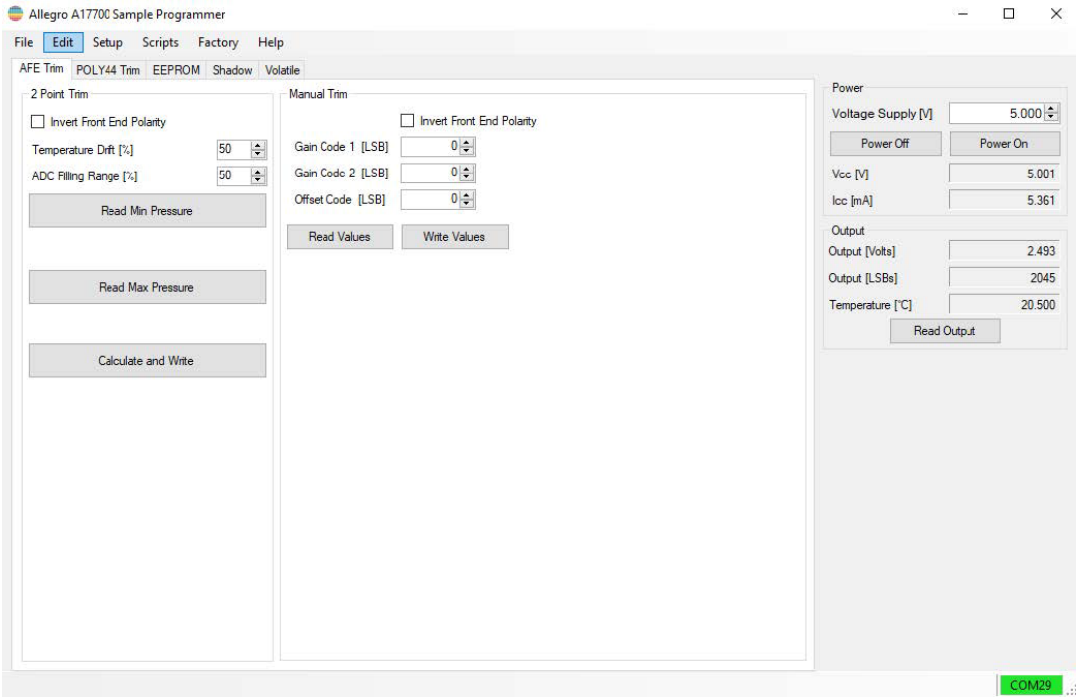

Figure 5: Allegro A17700 Sample Programmer GUI, "AFE Trim" Tab

Digital Polynomial

The A17700 features a 2D-polynomial correction block of 4th order in both temperature and pressure variables. This polynomial has proven outstanding performances on strain gauges. The following figures show the transfer function of the polynomial block. Note that the A17700 can also be used with lower degree polynomials; contact Allegro for further information.

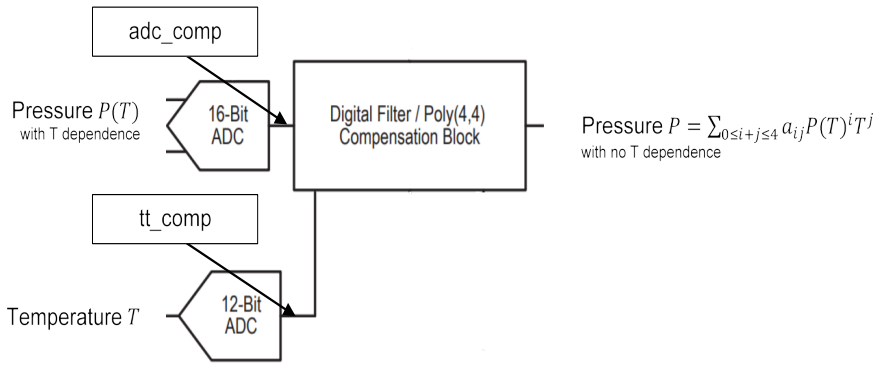

Figure 6: Digital Polynomial Compensation block inputs (Using bridge signal and temperature)

The polynomial compensation block offers the possibility to map the input duet, adc_comp and tt_comp, to an output pressure signal which is independent of the temperature.

Digital Polynomial – Data Acquisition

In order for the polynomial compensation to remove the temperature dependence of the pressure signal from the bridge, it is important to understand how this signal behaves versus applied pressure and applied temperature. The sampling of the characteristic displayed in Figure 1 is the most common method and is utilized herein. A higher number of samples directly improves the performance of the interface; the use of a minimum of 16 data points or a reconstruction of 16 data points through the reference bridge function is recommended.

The following table shows pressures and temperatures at which samples of adc_comp and tt_comp are acquired for this example where 16 calibration points are used to deduce the coefficients of a Poly(4,4) compensation.

Table 3: Example of Data Acquired During Digital Polynomial Compensation

Pressure	Temperature	adc_comp			tt_comp		
[Bar]	[°C]	16'h	Unsigned	Double	12'h	Unsigned	Double
0.02	–40	D269	53865	–0.356170654	EC0	3776	–40
119.8	–40	FB5E	64350	–0.036193848	EC0	3776	–40
240.1	–40	241D	9245	0.28213501	EC0	3776	–40
350.1	–40	49AF	18863	0.575653076	EC0	3776	–40
0.02	19	D1D6	53718	–0.360656738	098	152	19
120	19	F29A	62106	–0.104675293	098	152	19
239.9	19	1349	4937	0.150665283	098	152	19
350.6	19	31A1	12705	0.38772583	098	152	19
0.03	86.75	D162	53602	–0.364196777	2B6	694	86.75
120.17	86.625	EBDD	60381	–0.157318115	2B5	693	86.625
240.6	86.625	0653	1619	0.049407959	2B5	693	86.625
354.2	86.625	1ED7	7895	0.240936279	2B5	693	86.625
0.03	151.125	D13A	53562	–0.36541748	4B9	1209	151.125
121.9	150.875	E7C6	59334	–0.18927002	4B7	1207	150.875
240.3	150.75	FDA9	64937	–0.018280029	4B6	1206	150.75
351.4	150.625	1246	4678	0.14276123	4B5	1205	150.625

This step requires the access to the following memory registers and associated conversion:

Table 4: Digital EEPROM Information Used During Data Acquisition

Name	Address	Bits	Conversion from Unsigned to Double
adc_comp	0x52	[15:0]	**if** adc_comp >2^{15}–1 **then** adc_comp = (adc_comp – 2^{16})/2^{15} **else** adc_comp = adc_comp/2^{15} **return** adc_comp

Table 4: Digital EEPROM Information Used During Data Acquisition

Name	Address	Bits	Conversion from Unsigned to Double
tt_comp	0x62	[11:0]	**if** tt_comp >2^{11}–1 **then** tt_comp = (tt_comp – 2^{12})/2^{11} **else** tt_comp = tt_comp /2^{11} **return** tt_comp

Digital Polynomial – Coefficients Determination

The A17700 uses an internal DSP to finely adjust offset and sensitivity and to compensate for nonlinearity over temperature. The compensation algorithm takes as inputs the pressure as sensed by the external pressure sensor (Wheatstone bridge) and the temperature as sensed by the internal temperature sensor. The DSP compensates the pressure with a polynomial function up to 4th order that also includes temperature. Note that a_{00} is implicitly used to fine adjust the Offset and a_{10} implicitly used to fine adjust the Sensitivity.

$f(P, T) = a_{00} + a_{10}.p + a_{01}.t + a_{20}.p^2 + a_{11}.p.t + a_{02}.t^2 + a_{30}.p^3 + a_{21}.p^2.t + a_{12}.p.t^2 + a_{03}.t^3 + a_{40}.p^4 + a_{31}.p^3.t + a_{22}.p^2.t^2 + a_{13}.p.t^3 + a_{04}.t^4$

The coefficients are determined as a solution to a bounded least square equation from an input matrix "A" ("A" using acquired data from table 3) and a desired output "b", which is written as:

$$\min\|Ax - b\| \text{ such that } Ax \leq b \text{ and } l < x < u$$

where:

$$A = \begin{bmatrix} 1 & p_1 & t_1 & p_1^2 & p_1 \cdot t_1 & t_1^2 & p_1^3 & p_1^2 \cdot t_1 & p_1 \cdot t_1^2 & t_1^3 & p_1^4 & p_1^3 \cdot t_1 & p_1^2 \cdot t_1^2 & p_1 \cdot t_1^3 & t_1^4 \\ 1 & p_2 & t_2 & p_2^2 & p_2 \cdot t_2 & t_2^2 & p_2^3 & p_2^2 \cdot t_2 & p_2 \cdot t_2^2 & t_2^3 & p_2^4 & p_2^3 \cdot t_2 & p_2^2 \cdot t_2^2 & p_2 \cdot t_2^3 & t_2^4 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

A is the input matrix to the polynomial equation

p is adc_comp vector in double format

t is tt_comp vector in double format

every exponentiation or product is elementwise

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \end{bmatrix}$$

$$b = 0.0625 \cdot \left(2 \cdot \left(\alpha \cdot \frac{P - P_{MIN}}{P_{MAX} - P_{MIN}} + \beta\right) - 1\right);$$

P is the calibration pressure vector in bar

P_{MIN} and P_{MAX} are the limit of the operating range in bar, as defined in Figure 2

$\alpha = (V_{MAX} - V_{MIN})/4.6$ or equivalently $\alpha = (C_{MAX} - C_{MIN})/6279$

$\beta = (V_{MIN} - 0.2)/4.6$ or equivalently $\beta = (C_{MIN} - 956)/6279$

broadcasting is assumed when doing operation between scalar and vectors

$$l = [-1 \quad -0.9375 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1 \quad -1]$$

l is the minimum coefficient value

$$u = [1 \quad 1.0625 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$

u is the maximum coefficient value

Several numerical packages are solving these bounded least square equations:

- Matlab https://www.mathworks.com/help/optim/ug/lsqlin.html
- Alglib https://www.alglib.net/optimization/boundandlinearlyconstrained.php
- Python https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.optimize.lsq_linear.html

A MATLAB implementation is presented in the Appendix, but if use of computation software is not possible, Allegro MicroSystems also provide a complete solution including device programmer (ASEK-20), acquisition and computation GUI to complete steps described in this document.

The coefficients are gathered in the solution vector x. The mapping to the coefficients must be performed as presented in the following table. Note that a constant is additionally subtracted for the a10 coefficient. For the example, V_{MAX}=4.5 V, P_{MAX}=350 bar, V_{MIN}=0.5 V, P_{MIN}=0 bar were used.

Table 5: Digital EEPROM Information Used During Coefficients Determination

x	Coefficients			
Double	**Double**	**Unsigned**	**16'h**	**Name**
–0.00549	–0.00549	65356	FF4C	press_coeff_a00
0.110931	0.110931–0.0625= 0.048431[1]	1587	633	x[1]–0.0625= press_coeff_a10[1]
0.006348	0.006348	208	D0	press_coeff_a01
–0.00174	–0.00174	65479	FFC7	press_coeff_a20
–0.00583	–0.00583	65345	FF41	press_coeff_a11
0.073883	0.073883	2421	975	press_coeff_a02
0.126862	0.126862	4157	103D	press_coeff_a30
0.021667	0.021667	710	2C6	press_coeff_a21
–0.01764	–0.01764	64958	FDBE	press_coeff_12
0.000732	0.000732	24	18	press_coeff_a03
–0.00436	–0.00436	65393	FF71	press_coeff_a40
0.002747	0.002747	90	5A	press_coeff_a31
–0.00281	–0.00281	65444	FFA4	press_coeff_a22
–0.00278	–0.00278	65445	FFA5	press_coeff_13
–0.00198	–0.00198	65471	FFBF	press_coeff_04

[1] For this coefficient, an additional calculation step of subtracting 0.0625 from the *x* value is necessary.

This step requires the access to the following memory registers and associated conversion:

Table 6: EEPROM Information Used During Digital Polynomial Calibration

Name	Address	Bits	Conversion from Double to Unsigned
press_coeff_axx	0x0C to 0x15 See datasheet	See datasheet	press_coeff_axx = press_coeff_axx × 2^{15} if press_coeff_axx < 0 then press_coeff_axx = press_coeff_axx + 2^{16} else press_coeff_axx return press_coeff_axx

Strain Gauge and Calibrated A17700 Pressure Interface

The pressure module is now trimmed and can be evaluated to verify its functional behavior.

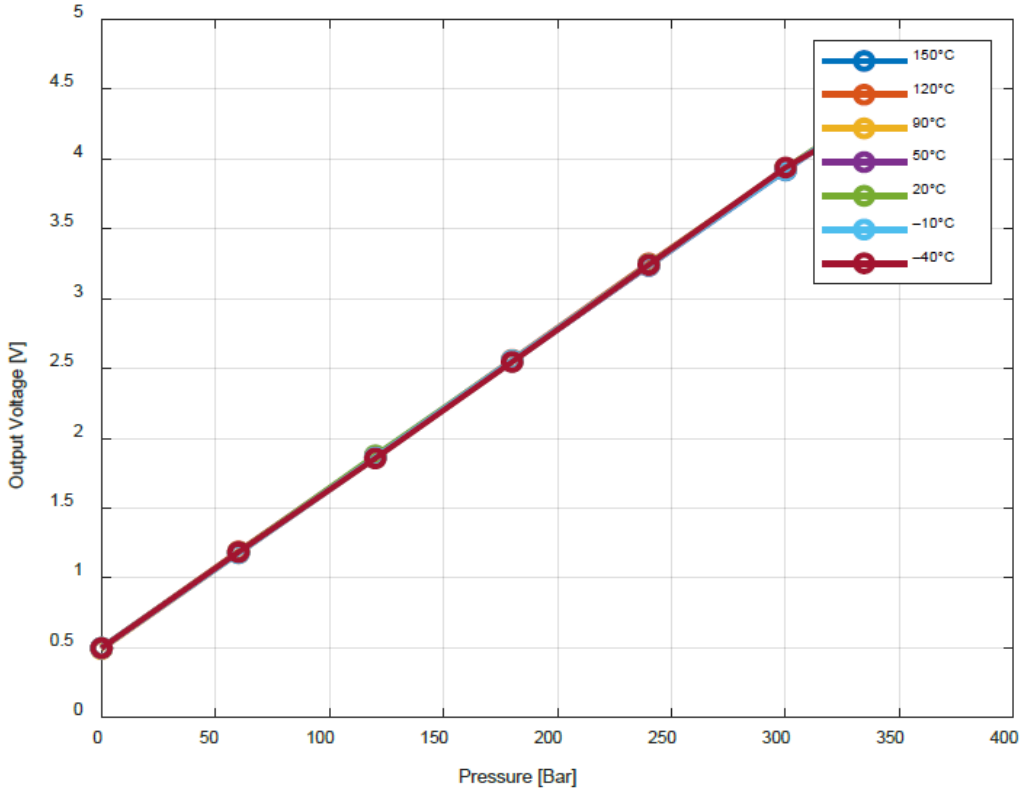

Figure 7: Piezoresistive Bridge output through A17700 Sensor Interface post coefficient programming

This verification step showcases the A17700 has been successfully programmed and temperature drift has been removed. The accuracy of the system has significantly improved and now reaches 0.35% of the full-scale output.

Full Compensation Flow

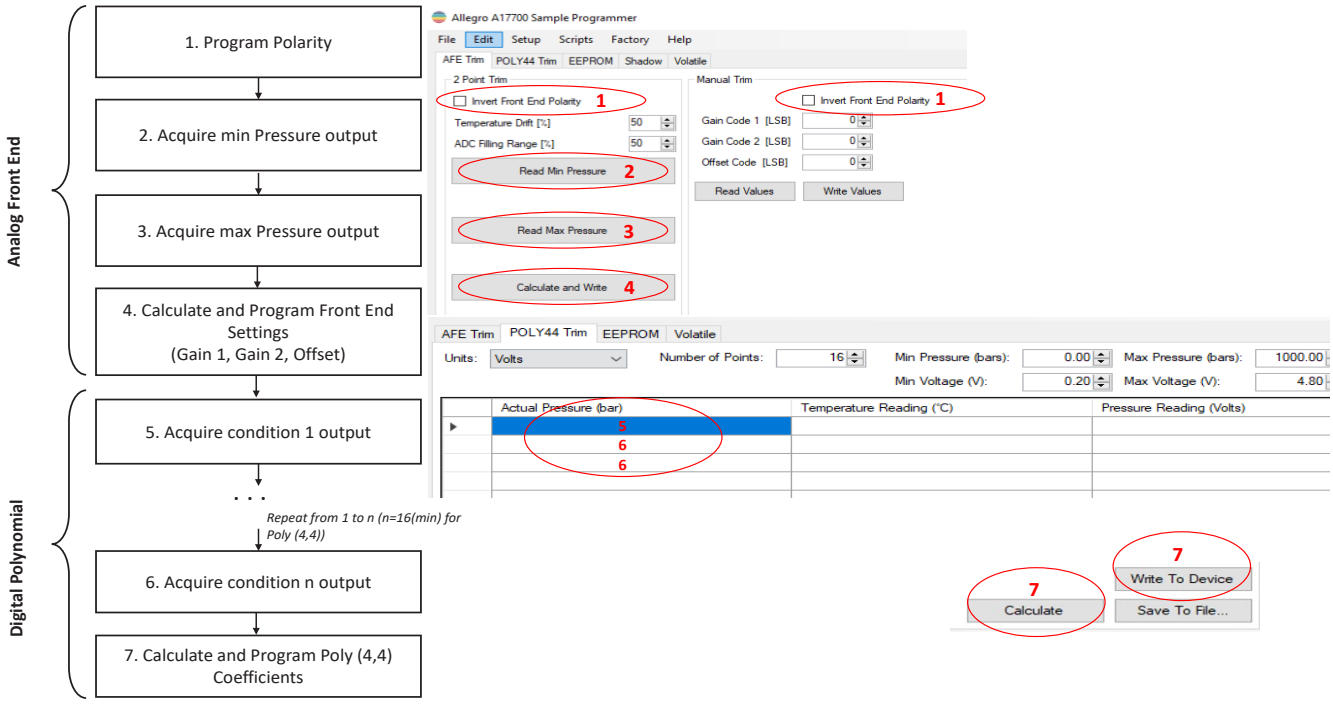

Figure 8: A17700 Compensation Flowchart and Equivalent Function On Allegro's A17700 Programming GUI

CONCLUSION

Pressure sensor interfaces are decisive tools to reach performant pressure sensing. The differential signal generated out of the pressure sensor bridge needs to be amplified and linearized over pressure and temperature. When correctly trimmed, Allegro's A17700 perfectly accomplishes this function allowing to take best benefit of the sensing element. This document provides a complete guide to compensation of output over pressure and temperature.

Allegro MicroSystems also provides a programming kit (ASEK-20) and a GUI to make data acquisition, calculate and program AFE code and Poly(4,4) coefficients to the A17700 interface, using the same method as described in this document.

APPENDIX: MATLAB IMPLEMENTATION

```
function
[Poly44_coeff,Error_estimated_perc_fs]=POLY44_Get_Coeff_bounded(adc_comp_reg,tt_comp_reg,calib_press_bar,max_operating_press_bar,min_operating_press_
bar,max_output_voltage,min_output_voltage,npoints)
% INPUTS
% ******
% - adc_comp_reg: (32x1) values read from adc_comp register ([15:0]@0x52) completed with zeros for row indexes>npoints [double]
% - tt_comp_reg : (32x1) values read from tt_comp register ([11:0]@0x62)
% completed with zeros for row indexes>npoints [double].
% - calib_press_bar : (32x1) calibration pressure values in bar [double]
% - max_operating_press_bar: (1x1) maximum operating pressure in bar
% [double].
% - min operating_press_bar: (1x1) minimum operating pressure in bar
% [double]
%- max_output_voltage= max output value in V (reached at
%max_operating_press_bar)
%- min_output_voltage= min output value in V (reached at
%operating_press_bar)
%- npoints: number of calibration points
%
% OUTPUTS
% *******
% - Poly44_coeff: structure of poly44 coefficients [double]
% - Error_estimated_perc_fs: estimation of the max error on the calibration
% points after trimming
%
% NOTE
% ****
% - Order of inputs must be consistent


if npoints>32
    npoints=32;
end
one_vec=zeros(32,1);
one_vec(1:npoints)=1;


% design matrix
A=[one_vec, adc_comp_reg, tt_comp_reg, adc_comp_reg.^2, adc_comp_reg.* tt_comp_reg, tt_comp_reg.^2, adc_comp_reg.^3, adc_comp_reg.^2.* tt_comp_reg,
adc_comp_reg.* tt_comp_reg.^2, tt_comp_reg.^3, adc_comp_reg.^4, adc_comp_reg.^3.* tt_comp_reg, adc_comp_reg.^2.* tt_comp_reg.^2, adc_comp_reg.* tt_comp_
reg.^3, tt_comp_reg.^4];

% target vector
alpha=(max_output_voltage-min_output_voltage)/4.600; %4600 is 0.92*5000
beta=(min_output_voltage-0.200)/4.600;  %4600 is 0.96*5000; 2000.0.04*5000;
b=2*(alpha*(calib_press_bar-min_operating_press_bar)/(max_operating_press_bar-min_operating_press_bar)+beta)-1;
b(npoints+1:end)=0;
b=b*0.0625;

% parameter range
l=-1*ones(15,1);
u=1*ones(15,1);
l(2)=l(2)+0.0625;
u(2)=u(2)+0.0625;


% fitting
[x,~,error]=lsqlin(A,b,zeros(1,15),Inf,zeros(1,15),0,l,u);
Error_estimated_perc_fs=100*max(abs(error))/(max(b)-min(b));
x(2)=x(2)-0.0625;

Poly44_coeff.a00 = x(1);
Poly44_coeff.a01 = x(3);
Poly44_coeff.a02 = x(6);
Poly44_coeff.a03 = x(10);
Poly44_coeff.a04 = x(15);
Poly44_coeff.a10 = x(2);
Poly44_coeff.a11 = x(5);
Poly44_coeff.a12 = x(9);
Poly44_coeff.a13 = x(14);
Poly44_coeff.a20 = x(4);
Poly44_coeff.a21 = x(8);
Poly44_coeff.a22 = x(13);
Poly44_coeff.a30 = x(7);
Poly44_coeff.a31 = x(12);
Poly44_coeff.a40 = x(11);

end
```

Revision History

Number	Date	Description	Responsibility
–	June 29, 2021	Initial release	Cedric Gillet

955 PERIMETER ROAD • MANCHESTER, NH 03103 • USA
+1-603-626-2300 • FAX: +1-603-641-5336 • ALLEGROMICRO.COM