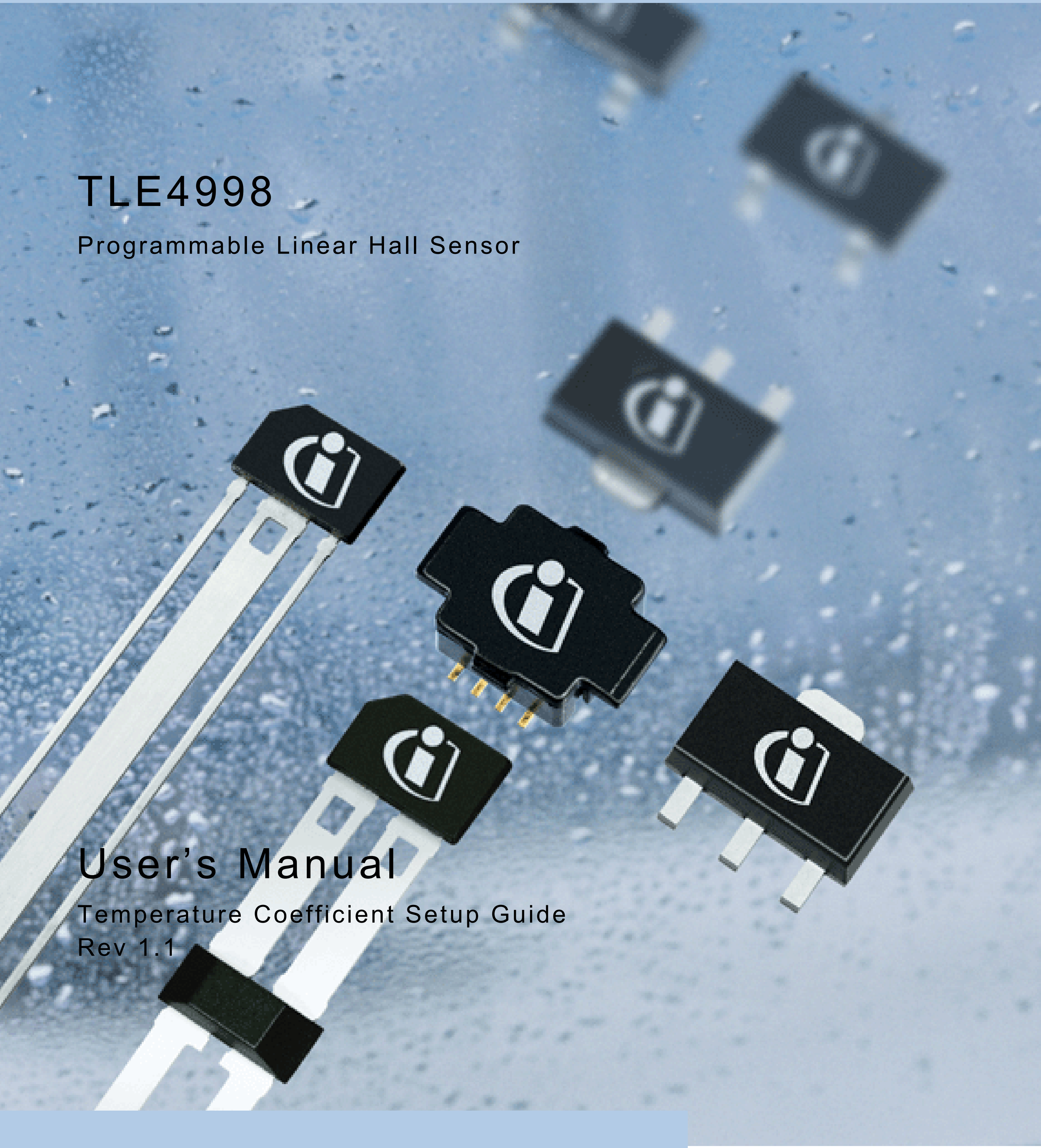
October 2008



TLE4998

Programmable Linear Hall Sensor

User's Manual

Temperature Coefficient Setup Guide

Rev 1.1

Sense & Control

Never stop thinking

Edition 2008-10-30

Published by
Infineon Technologies AG
81726 Munich, Germany

TLE4998 Programmable Linear Hall Sensor

Revision History: 2008-10-30, Rev 1.1

Previous Version: Rev 1.0	
Page	**Subjects (major changes since last revision)**
7	Changes in **Table 1**: TCAL, HCAL calculations
9	Corrected junction temperature calculation
10	Included offset and normalization in user sensitivity calculation in **Equation (9)** and **Equation (10)**
12	Presented more straightforward rms error criterion in **Equation (15)**
12	Adapted Visual Basic implementation example (conforming names, lean structure)

Table of Contents

1 Introduction

This document is a supplement to the TLE4998 datasheet.

- It is valid for all TLE4998 products and derivatives
- It is intended as add-on to the currently available TLE4998 target and/or data sheets
- It is recommended to read the programming description of this device before
- It gives an overview and detailed description of the temperature compensation concept of the TLE4998

1.1 Concept of the TLE4998 Temperature Compensation

The concept is based on shifting as much of the on-chip sensor processing as possible to the digital domain. This can be achieved by directly converting the output of the Hall probe into a digital value, without additional, purely analog pre-processing. The magnetic range setup is also done directly in the first stage of the sigma-delta analog-to-digital converter (Hall ADC) used in the TLE4998. We call this principle magnetic-to-digital conversion (MDC).

By doing that, and of course by using a Hall ADC with outstanding performance, the compensation requirements are mainly reduced to the magnetic circuit of the application and the Hall probe itself. All other parts do no longer have any significant temperature depedency. This principle requires an accurate junction temperature value in digital form for calculation of compensation polynomials. Accordingly, an accurate temperature ADC is included in the sensor as well. **Figure 1** shows a simplified block diagram of the TLE4998 to demonstrate its temperature compensation concept.

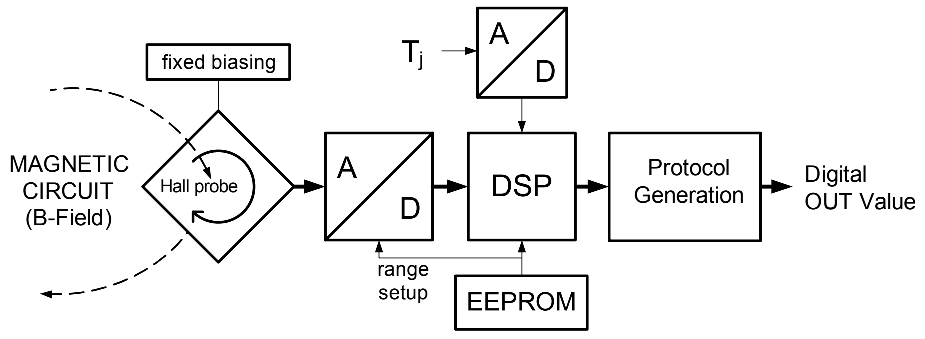


Figure 1 Simplified Block Diagram of the TLE4998

1.2 Signal Processing

Figure 2 shows the signal flow diagram including important internal data values. The values in green boxes can be programmed in EEPROM. A detailed description on how to access and write those values is given in the TLE4998 user programming guide. **Figure 3** shows the main data processing steps of the TLE4998 digital part. As illustrated, the processing of the Hall-ADC update occurs with typically 16k SPS (samples per second). The DSP data processing has to be performed according to the selected Interface mode. For SENT, the processing of a new output value takes place during the synchronisation frame, for PWM the processing is done before the next following frame.

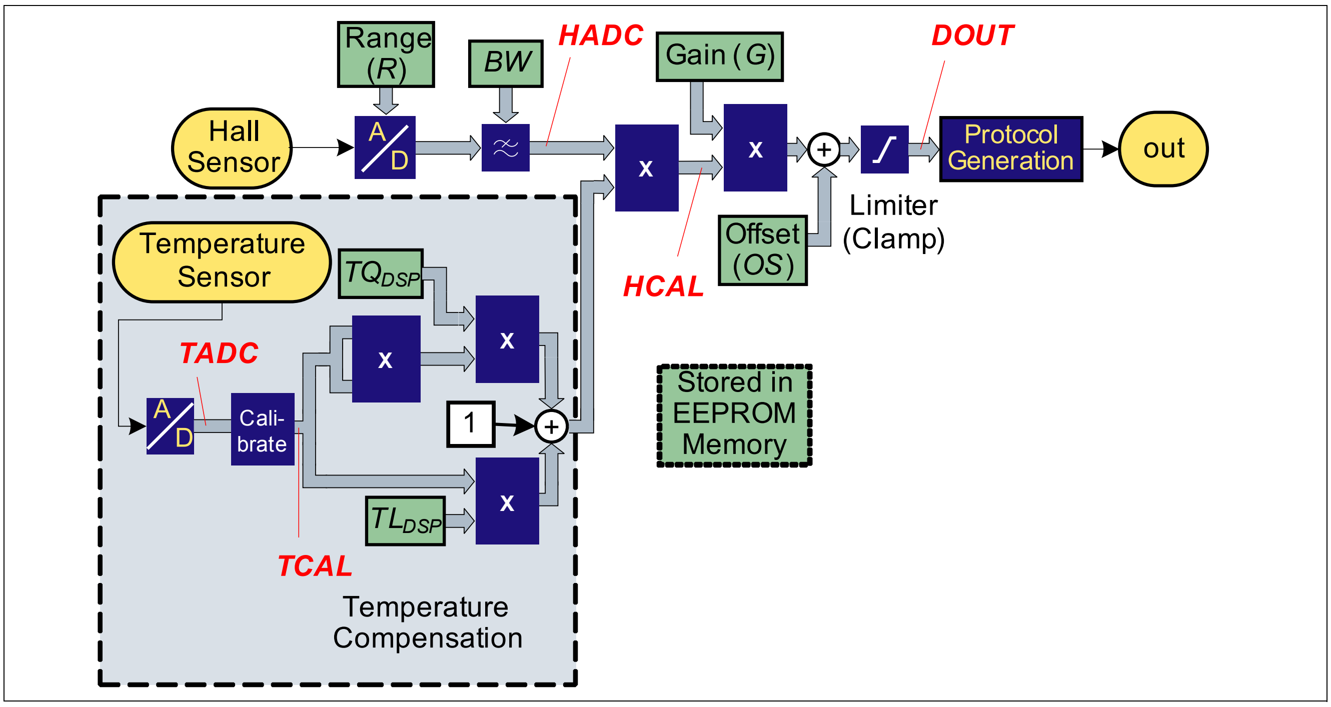


Figure 2 Block Diagram of the TLE4998

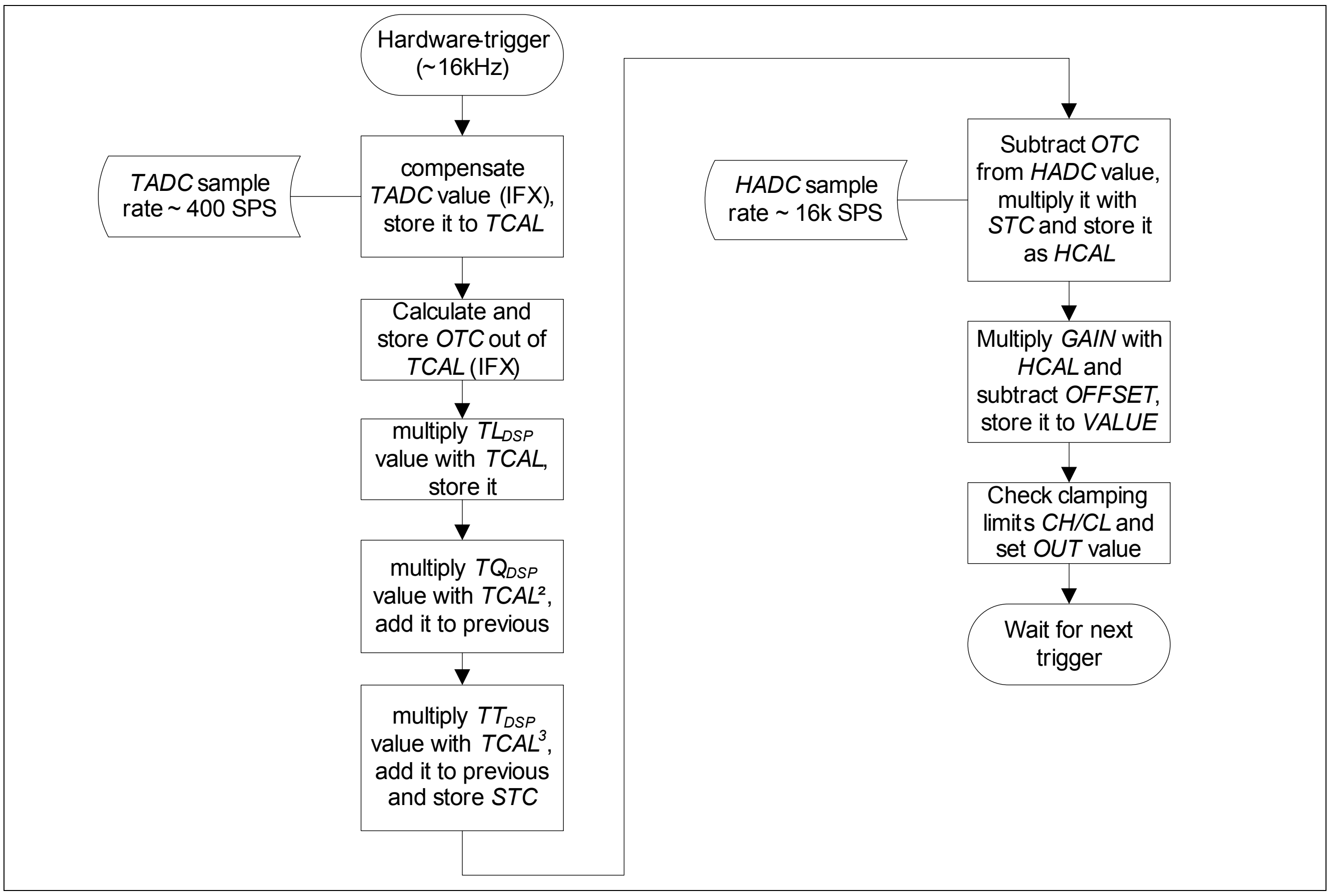

Hardware-trigger (~16kHz)
TADC sample rate ~ 400 SPS
compensate TADC value (IFX), store it to TCAL
Calculate and store OTC out of TCAL (IFX)
multiply TLDSP value with TCAL, store it
multiply TQDSP value with TCAL², add it to previous
multiply TTDSP value with TCAL³, add it to previous and store STC
HADC sample rate ~ 16k SPS
Subtract OTC from HADC value, multiply it with STC and store it as HCAL
Multiply GAIN with HCAL and subtract OFFSET, store it to VALUE
Check clamping limits CH/CL and set OUT value
Wait for next trigger

Figure 3 Data Processing Flowchart

2 Mathematical Background

To describe the overall temperature dependent behaviour of the IC, we need to consider both the calculated DSP compensation as well as the natural (physical) temperature behaviour of the Hall probe. **Table 1** gives some more detailed insights in the algorithm that is performed in the TLE4998. Please note that this document treats the temperature compensation only. The determination of the two parameters *GAIN* and *OFFSET* is explained in the 2 point calibration setup document.

Table 1 Equations performed in the DSP

Internal Value (16bit)	**Calculation (20bit integer operations)**
TCAL	$16T_j - 768$ (T_j is calibrated junction temperature in °C)
OTC	IFX calibrated (and T dependent) offset value
STC1	$8192 + ((TL_{DSP} - 160)TCAL) \gg 7$
VALUE1	$(TCAL \cdot TCAL) \gg 10$
STC2	$((TQ_{DSP} - 128)VALUE1) \gg 8 + STC1$
VALUE2	$(TCAL \cdot VALUE1) \gg 11$
STC	$(-TT_{DSP} \cdot VALUE2) \gg 6 + STC2$
HCAL	$(((HADC + OTC) \cdot STC) \gg 13) \cdot 3 \gg 1$
VALUE	$((G - 16384) \cdot HCAL) \gg 11 + (OS - 16384) \ll 4 + 32768$
DOUT	*VALUE* (but limited to given *CL* and *CH* values)

The sensitivity STC can then be written as a function of *TCAL:*

$$STC(TCAL) = 8192 + \frac{TL_{DSP} - 160}{8}\left(\frac{TCAL}{16}\right) + \frac{TQ_{DSP} - 128}{1024}\left(\frac{TCAL}{16}\right)^2 - \frac{TT_{DSP}}{32768}\left(\frac{TCAL}{16}\right)^3 . \qquad (1)$$

To get a normalized DSP sensitivity S_{DSP}, the value needs to be normalized by dividing by 8192. We then have

$$S_{DSP}(T_j) = 1 + \frac{TL_{DSP} - 160}{8 \cdot 8192}\left(\frac{TCAL}{16}\right) + \frac{TQ_{DSP} - 128}{1024 \cdot 8192}\left(\frac{TCAL}{16}\right)^2 - \frac{TT_{DSP}}{32768 \cdot 8192}\left(\frac{TCAL}{16}\right)^3 \qquad (2)$$

with the junction temperature being expressed as $T_j = TCAL/16 + 48$.

2.1 Influence of the DSP

All processing in the DSP happens in a deterministic way using integer operation, which has to be kept in mind when reconstructing its behaviour in a simulation model (Otherwise, when using floating point operations, some notifiable truncation effects could lead to small inconsistencies between the model and the reality). Nevertheless, the system is designed in a way that this error should be always less than one LSB for the resulting digital output value.

Therefore, it is necessary to know that the calculation of one line shown in **Table 1** uses 20bit (signed) integers and that the results itself are stored in 16bit (signed) integers. All combined “multiply and shift right” operations use virtually a broader integer width (20bit plus the amount of bits shifted right) due to the employed mechanism in the processor. When storing a value, a proper saturation is taking place, so that there is no clipping for too big/small results of any calculation. When passing a value into another equation, a proper sign extension is performed.

2.2 Influence of the Hall Probe

The Hall probe has a higher order behaviour over temperature. Generally, it can be approximated by a sensitivity

$$S_{probe}(T_j) = 1 + TC1_{probe}T_j + TC2_{probe}T_j^2 + TC3_{probe}T_j^3, \tag{3}$$

where T_j is the junction temperature, represented by the $TCAL$ value used by the DSP, and $TC1_{probe}$, $TC2_{probe}$ and $TC3_{probe}$ are the linear, quadratic and cubic temperature coefficients of the Hall probe, respectively.

2.3 Overall Behaviour of the IC: The Precalibration at IFX

With the knowledge of the behaviour of DSP and Hall probe, the complete compensated system can be described as

$$S_{pre}(T_j) = S_{probe}(T_j) \cdot S_{DSP,\,pre}(T_j), \tag{4}$$

where the subscript *pre* refers to precalibration. Inserting **Equation (2)** (the DSP behaviour) and **Equation (3)** (the Hall probe behaviour) into **Equation (4)** (having $TCAL = 16T_j - 768$), we get

$$S_{pre}(T_j) = (1 + TC1_{probe}T_j + TC2_{probe}T_j^2 + TC3_{probe}T_j^3) \left(1 + \frac{TL_{DSP,\,pre} - 160}{8 \cdot 8192}\left(\frac{TCAL}{16}\right) + \frac{TQ_{DSP,\,pre} - 128}{1024 \cdot 8192}\left(\frac{TCAL}{16}\right)^2 - \frac{TT_{DSP,\,pre}}{32768 \cdot 8192}\left(\frac{TCAL}{16}\right)^3\right) \tag{5}$$

This means that during precalibration at Infineon the parameters TR_{DSP}, TL_{DSP} and TQ_{DSP} are programmed (precalibrated, therefore the subscript *pre*) to reach an overall “flat” temperature behaviour (constant 100% of sensitivity over the whole temperature range, $S_{pre}(T_j) \cong 1$). This is done by acquiring the probe sensitivity polynomial and calculating its inverse to determine the DSP polynomial.

The measurement during precalibration has a finite accuracy in terms of the applied magnetic fields and temperatures. This leads to a certain (but small) sensitivity error residue over temperature. Also, the DSP parameters have some finite quantization. Finally, some minor influences of other parts we neglected might sum up (e.g. the accuracy of the on-chip temperature). These factors are the main cause for the not 100% perfect calibration.

2.4 Calibration of the Application Circuit

The last step is now to compensate the external magnetic circuitry. Every magnet has some variation in its field strength over temperature and additional effects of the application (such as thermal expansions) may contribute to a non-constant magnetic field at one particular position if measured at different temperatures. Modeled as a sensitivity with respect to temperature changes, we call this application behaviour $S_{app}(T)$. The TLE4998 system allows the user to compensate those effects via a second-order user polynomial

$$S_{user}(T) = 1 + TC1_{user}(T - T0_{user}) + TC2_{user}(T - T0_{user})^2, \tag{6}$$

where the three parameters $T0_{user}$ (Reference temperature for a sensitivity of 1), $TC1_{user}$ (linear user temperature coefficient), $TC2_{user}$ (quadratic user temperature coefficient) are user selectable. The temperature T is somewhat lower than the junction temperature T_j, as the sensor die has a certain power dissipation and the package a specific thermal resistance R_{thJA}. You may want to include this difference later to improve the setup, for understanding the principle (based on these explanations) it is not required. Thus, we define $T = T_j = T_a$.

Again, we are looking for an overall flat response over temperature, cancelling out all the effects of the application circuit. Therefore, we have to chose the user parameters $T0_{user}$, $TC1_{user}$ and $TC2_{user}$ in a way that the user and application sensitivity cancel out:

$$S_{user}(T) = \frac{1}{S_{app}(T)}. \tag{7}$$

The overall system response then consists of the following multiplication of sensitivities:

$$S_{overall}(T) = S_{app}(T) \cdot S_{user}(T) \cdot S_{probe}(T) \cdot S_{DSP,\,pre}(T). \tag{8}$$

Chapter 3 shows how to measure the application sensitivity $S_{app}(T)$ and how to get the user parameters from there.

3 TC Setup Part 1 - Obtaining the Magnetic Circuit Behaviour

As explained in the previous chapter, precalibrated TLE4998 devices delivered by Infineon have a flat behaviour over temperature. So it is possible to use the device itself to determine the temperature behaviour of the application circuitry. Furthermore, the required temperature values can be acquired directly, too, without the need for an external temperature measurement setup plus complicated calculations of differences between chip/junction and application/ambient temperatures.

The idea is to use a certain number of reference application setups and measure their temperature behaviour. This leads to a certain average temperature behaviour and also some statistics of its distribution. These evaluation results are then used at production for directly programming the sensor ICs without individual measurements at specific temperatures.

3.1 Measurement Setup

We assume a magnetic circuit setup with a TLE4998 device as illustrated in **Figure 4**.

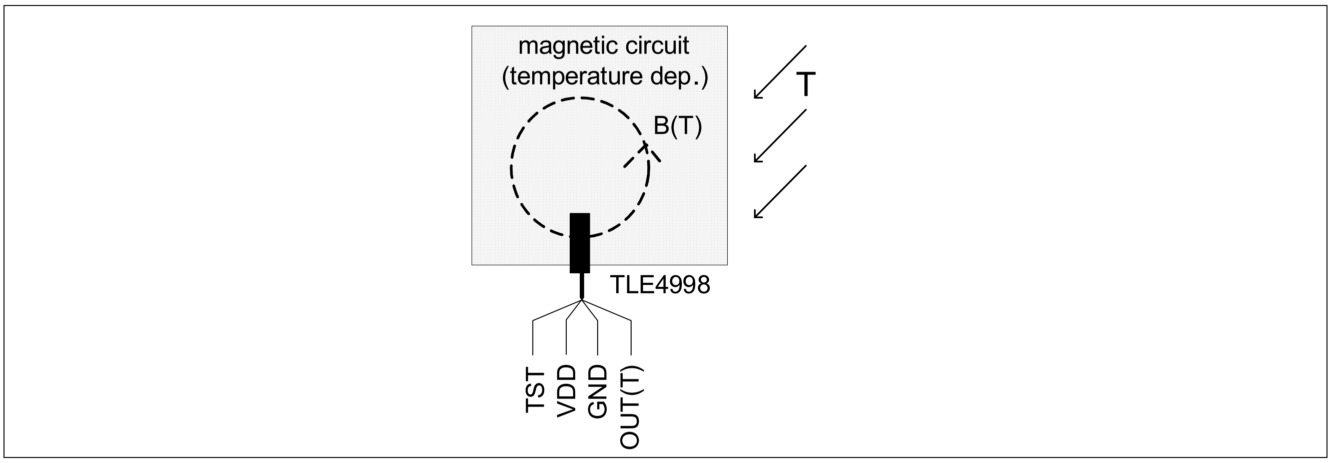


Figure 4 Principle of a Magnetic Circuit Setup with Temperature Dependency

A decoding unit (oscilloscope, microcontroller) for the digital output value is required. Besides that, the temperature applied must be measured either by an external temperature device (this is T_a - in that case the corresponding junction temperature T_j must be mathematically determined, see TLE4998 specification for details) or by using the programming interface of the TLE4998. In the latter case, the junction temperature *TCAL* used by the sensor is directly given as decimal value in degrees Celsius. To get the real junction temperature, please calculate $T_j = TCAL/16 + 48$ (see programming description for details).

When using the PGSISI programmer/demonstration kit for the TLE4998, the digital output value is directly measured and the temperature is correctly read and displayed. With this configuration no further measurement equipment is required, but as it is a low cost device (for a first evaluation only) it does not (and can not) unveil the full performance of the TLE4998.

3.2 Finding the User Coefficients to Compensate the Magnetic Circuit Behaviour

With the given measurement setup it is possible to acquire the application sensitivity behaviour by measuring the sensor output $OUT_B(T)$ for a point with non-zero magnetic field at at least three different temperatures T_k while keeping the other parameters constant. We also have to take into account the zero field output of the sensor OUT_0 and divide by it to cancel out the units. The subscript *unnorm* points out that we don't yet want to normalize this sensitivity to be exactly unity (it will be done in the next step):

$$S_{app,\,unnorm}(T_k) = \frac{OUT_B(T_k) - OUT_0}{OUT_0} \tag{9}$$

Considering **Equation (7)**, the user polynomial $S_{user,\,unnorm}(T_k)$ can then be found as:

$$S_{user,\,unnorm}(T_k) = \frac{OUT_0}{OUT_B(T_k) - OUT_0} \tag{10}$$

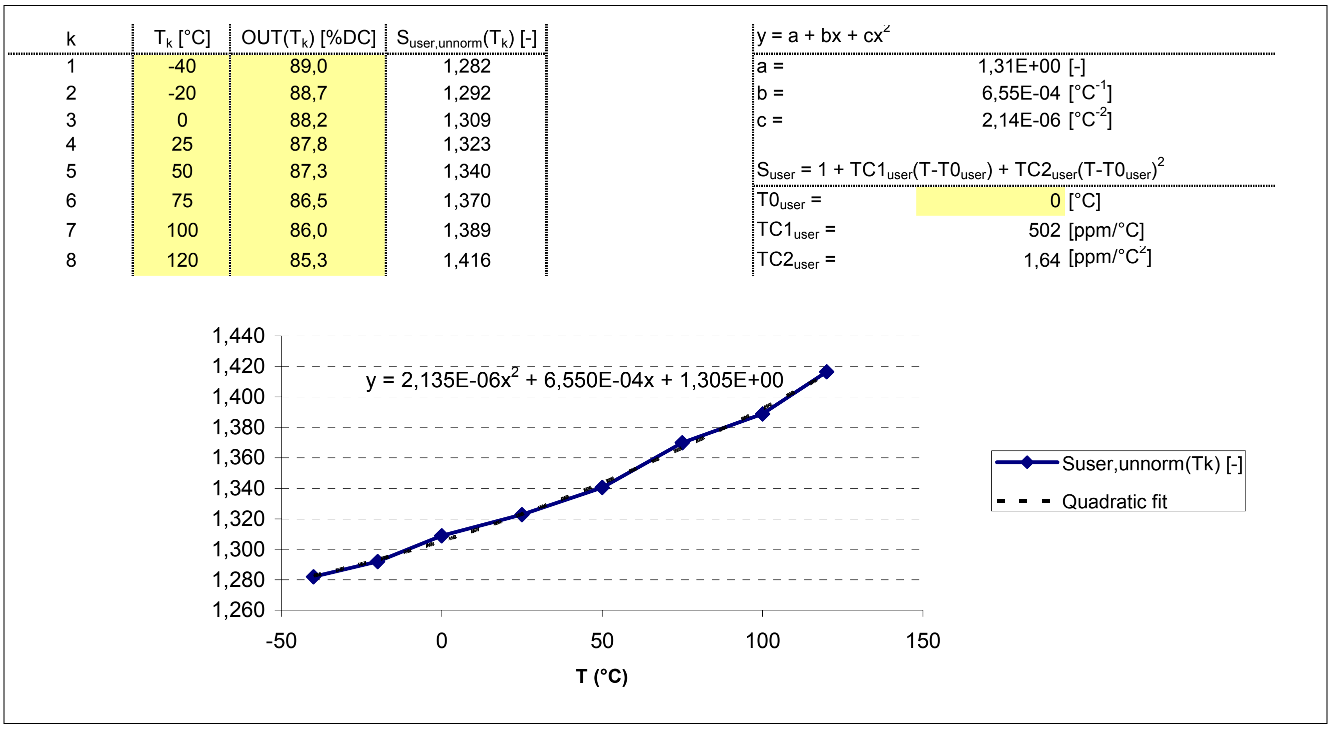
We can then generate a quadratic polynomial fit using the least squares error method or other approaches (Please check any standard math textbook for this task). An easy way is using Microsoft® Excel®, enter the points $S_{user,\,unnorm}(T_k)$, generate an XY-graph out of this dataset and use the trendline function to get a 2nd order polynomial fit. Excel can also display the function of the second order fit in the form $y(x) = a + bx + cx^2$. By comparing this to **Equation (6)** ($S_{user}(T) = 1 + TC1_{user}(T - T0_{user}) + TC2_{user}(T - T0_{user})^2$), we can see that the easiest set of user coefficients is found by choosing $T0_{user} = 0$ °C and normalizing the linear and quadratic terms by taking $TC1_{user} = b/a$ °C^{-1} and $TC2_{user} = c/a$ °C^{-2}. Using these settings leads to a unity sensitivity at 0 °C. The exact normalization is not very important because it will be corrected by the gain factor *G* factor during the subsequent two-point calibration. **Figure 2** gives an example how user parameters can be found based on a series of output values.

3.2.1 Special Case of Non-Zero TR$_{user}$

In the special case if the user reference temperature is not chosen as 0°C, the conversion from the Excel values is slightly less straightforward. The mapping between the user coefficients $TC1_{user}$, $TC2_{user}$ and the Excel parameters *a*, *b* and *c* can be obtained using the following formulas:

$$TC1_{user} = \frac{b + 2c \cdot T0_{user}}{a + b \cdot T0_{user} + c \cdot T0_{user}^{\;2}} \tag{11}$$

$$TC2_{user} = \frac{c}{a + b \cdot T0_{user} + c \cdot T0_{user}^{\;2}} \tag{12}$$



k	T_k [°C]	OUT(T_k) [%DC]	$S_{user,unnorm}(T_k)$ [-]
1	-40	89,0	1,282
2	-20	88,7	1,292
3	0	88,2	1,309
4	25	87,8	1,323
5	50	87,3	1,340
6	75	86,5	1,370
7	100	86,0	1,389
8	120	85,3	1,416

$y = a + bx + cx^2$		
a =	1,31E+00	[-]
b =	6,55E-04	[°C^{-1}]
c =	2,14E-06	[°C^{-2}]

$S_{user} = 1 + TC1_{user}(T-T0_{user}) + TC2_{user}(T-T0_{user})^2$		
$T0_{user}$ =	0	[°C]
$TC1_{user}$ =	502	[ppm/°C]
$TC2_{user}$ =	1,64	[ppm/°C^2]

Figure 5 Example of User Parameter Retrieval Using Microsoft Excel

4 TC Setup Part 2 - Calculating the New EEPROM Values

There is only one parameter set stored in the TLE4998 to calculate a single implemented polynomial. This means that the new user polynomial is not stored separately in the sensor, but needs to be combined with the precalibration polynomial. The new parameter set is then again stored back to the sensor. The reason is simply that the calculation must only be done once (at start of life). Further advantages are

- It saves required EEPROM bits for the second parameter set
- It improves reliability (easier protection of less “influencable” EEPROM bits)
- It reduces power consumption for the calculation of this second polynomial

The user sensitivity $S_{user}(T)$ must therefore be combined with the precalibration DSP sensitivity $S_{DSP,pre}(T)$ to find the final DSP sensitivity $S_{DSP,final}(T)$, which must be chosen such as to minimize the overall system deviation from a constant output over temperature. This means that we want to find $TL_{DSP,final}$ and $TQ_{DSP,final}$, which lead to

$$S_{DSP,final}(T) = Cst \cdot S_{DSP,pre}(T) \cdot S_{user}(T) \cdot (1 + \varepsilon(T)), \tag{13}$$

where a constant *Cst* is included, which is independent of temperature and can therefore be accounted for with the *GAIN* factor set during the two-point calibration and $\varepsilon(T)$ is the residual error of the calculation, which is a function of the temperature and has to be minimized. Due to the fact that a second order (user) and a third order polynomial (DSP) are multiplied, it is not possible to find an exact final DSP polynomial. Since the second and especially the third order term are comparably small, it is however possible to find sufficiently accurate parameters $TL_{DSP,final}$ and $TQ_{DSP,final}$, in the available parameter space to minimize the error $\varepsilon(T)$, which can be expressed as

$$\varepsilon(T) = \frac{S_{DSP,final}(T)}{Cst \cdot S_{DSP,pre}(T) \cdot S_{user}(T)} - 1. \tag{14}$$

Cst can for example be found by calculating the average error with *Cst* set to one as illustrated in the implementation example.

4.1 Algorithm

For a computing system, an iterative algorithm is a good way to solve for the new DSP parameters as the possible parameter space is very small. This means we iterate through all coefficients and check where **Equation (13)** is satisfied best over temperature. One possible criterion is to minimize the root mean square error value ε_{rms}, which is given by

$$\varepsilon_{rms} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(-1+\frac{S_{DSP,\,final}(T_k)}{Cst\cdot S_{DSP,\,pre}(T_k)\cdot S_{user}(T_k)}\right)^2}\,. \tag{15}$$

It is not necessary to determine the residual error for very small temperature steps, a good value is to look for the maximum residual error every 10°C from the minimum required temperature to the maximum required temperature (-40 °C to +150 °C are resonable limits). The following section illustrates the procedure on a practical implementation example.

4.2 Example Implementation in Visual Basic

The implementation is separated into two parts:

- Initialization of several global variables (parameters etc.)
- Subroutines for calculating all the different sensitivities
- The iteration loop to calculate the parameter set

The implementation uses Microsoft® Visual Basic®, this language should be easy to understand and to map into other programming languages.

4.2.1 Setup of Global Variables

Precalibrated parameter set read from the sensor:

- TL_dsppre (as stored in EEPROM)
- TQ_dsppre (as stored in EEPROM)
- TT_dsppre (as stored in EEPROM)

Valid ranges are: TL_dsppre: 0...511; TQ_dsppre: 0...255; TT_dsppre: 0...31.

Given user values in °C and ppm:

- T0_user (user reference temperature)
- TC1_user (user linear temperature coefficient)
- TC2_user (user quadratic temperature coefficient)

Valid ranges are: T0_user: -50...80, TC1_user: -0.001...0.0025, TC2_user: -0.000004...0.000004.

Last but not least we have the new setup values, we initialize partly:

- TL_dsp (used as sweep variable, needs no initialization)
- TQ_dsp (used as sweep variable, needs no initialization)
- TT_dsp = TT_dsppre

(Valid ranges are: TL_dsp: 0...511, TQ_dsp: 0...255, TT_dsp: 0...31).

4.2.2 Sensitivity Calculation Subroutines

The first function calculates the user polynomial at a given temperature T:

```
Private Function S_user(ByVal T) As Double
    S_user = 1 + TC1_user * (T - T0_user) + TC2_user * (T - T0_user) ^ 2
```

```
End Function
```

The next function calculates the sensor DSP behaviour after precalibration at a given temperature T:

```
Private Function S_dsppre(ByVal T) As Double
    S_dsppre = 1 + (TL_dsppre - 160) * (T - 48) / 8 / 8192
                     + (TQ_dsppre - 128) * ((T - 48) ^ 2) / 1024 / 8192
                     - TT_dsppre * ((T - 48) ^ 3) / 32768 / 8192
End Function
```

Finally, there is the calculation function for the DSP polynomial at a temperature T:

```
Private Function S_dsp(ByVal T) As Double
    S_dsp = 1 + (TL_dsp - 160) * (T - 48) / 8 / 8192 +
              (TQ_dsp - 128) * ((T - 48) ^ 2) / 1024 / 8192 -
              TT_dsp * ((T - 48) ^ 3) / 32768 / 8192
End Function
```

For the algorithm we need the already explained error function:

```
Private Function epsilon(ByVal T) As Double
    epsilon = -1 + S_dsp(T) / (Cst * S_dsppre(T) * S_user(T))
End Function
```

4.2.3 Iteration Loop

The iteration loop looks through all TL_dsp and TQ_dsp values and checks for a best fit (smallest rms error). It is done in two steps, first it looks for a global optimum with a stepsize of 10 for both parameters. In a second step it iterates again using stepsize 1 around the global optimum to find a local optimum.

```
Rem --> Initialize temperature sweep parameters
T_min = -40
T_max = 150
T_step = 10
n = Math.Round((T_max - T_min) / T_step)

Rem --> Initialize variables to keep track of current optimum values
epsilon_rms_opt = 9999
TL_opt = 0
TQ_opt = 0

Rem =========================================================================
Rem --> Sweep in two runs, the coarse global and the fine local search
For Rounds = 1 To 2

    Rem =====================================================================
    Rem --> Initialize sweep parameters
    Rem     First round coarse, second round fine
    If (Rounds = 1) Then
        TL_lo = 10
        TL_hi = 500
        TL_step = 10
        TQ_lo = 10
        TQ_hi = 240
        TQ_step = 10
    ElseIf (Rounds = 2) Then
        TL_lo = TL_opt - 9
        TL_hi = TL_opt + 9
        TL_step = 1
        TQ_lo = TQ_opt - 9
        TQ_hi = TQ_opt + 9
        TQ_step = 1
    End If

    Rem =====================================================================
    Rem --> TL sweep
    For TL_dsp = TL_lo To TL_hi Step TL_step

        Rem =================================================================
        Rem --> TQ sweep
        For TQ_dsp = TQ_lo To TQ_hi Step TQ_step

            Rem =============================================================
            Rem --> Determine minimizing Cst by calculating average epsilon
            epsilon_sum = 0
            Cst = 1
            For T = T_min To T_max Step T_step
                epsilon_sum = epsilon_sum + epsilon(T)
            Next
            epsilon_mean = (epsilon_sum / n)
            Cst = epsilon_mean + 1

            Rem =============================================================
            Rem --> Determine rms value of epsilons
            epsilon_sum = 0
            For T = T_min To T_max Step T_step
                epsilon_sum = epsilon_sum + epsilon(T) ^ 2
            Next
            epsilon_rms = Math.Sqr(epsilon_sum / n)

            Rem =============================================================
            Rem --> Determine if new optimum parameters were found
            If epsilon_rms < epsilon_rms_opt Then
                epsilon_rms_opt = epsilon_rms
                TL_opt = TL_dsp
                TQ_opt = TQ_dsp
            End If
        Next
    Next
Next

Rem =========================================================================
Rem --> Finally retrieve the best TL, TQ values stored during the sweep
TL_dspfinal = TL_opt
TQ_dspfinal = TQ_opt
```

At the end, only *TL_dspfinal* and *TQ_dspfinal* need to be reprogrammed in the EEPROM.

4.3 Summary of the Procedure

To program a new user temperature coefficient to sensors during production, perform following steps. Please note that not a single measurement or any further verification of the sensor temperature behaviour is required:

- Readout the precalibration parameters from the EEPROM
- Use this data plus the required user setup to calculate a new dataset
- Program the new EEPROM values