AN12301

MM9Z1x638 system calibration guidelines

Rev. 1 — 5 June 2019 Application note

Document information

Information	Content
Keywords	battery, BMS, sensor, IBS, calibration
Abstract	The purpose of this application note is to explain the overall concept of the calibration of the MM9Z1x638 device and to give guidelines on how to implement a system calibration.

Revision history

Rev	Date	Description
1	20190605	initial version

1 Introduction

The MM9Z1x638 intelligent battery sensor (IBS) is a high precision measurement device. NXP can only guarantee the performance on device level (on the package pins). In the final application, consider additional effects caused, e.g., by external components, printed-circuit board (PCB), device configuration, and the soldering process.

To achieve the high precision in the final application, implement a calibration on system level.

1.1 Conventions for register and bit referencing

The following convention references register names and bits.

REGISTERNAME.BITNAME(MASK)

Example:

ACQ_ACC1.TCOMP(M)

REGISTERNAME references the respective register, e.g. ACQ_ACC1 – acquisition chain control register 1.

BITNAME references the respective register bit, e.g. TCOMP bit in acquisition chain control register 1.

(M) indicates a mask bit, e.g. TCOMPM bit in acquisition chain control register 1.

Masked bits allow to set or clear individual bits without the need for a read-modify-write process. For example see the following code example:

```
/*
* Copyright 2016 - 2019 NXP, All rights reserved.
*
* SPDX-License-Identifier: BSD-3-Clause
*/

//enable (set) compensation for TSENSE channel
B_ACQ_ACC1 = B_ACQ_ACC1_TCOMPM_MASK| B_ACQ_ACC1_TCOMP_MASK;

//disable(clr) compensation for TSENSE channel
B_ACQ_ACC1 = B_ACQ_ACC1_TCOMPM_MASK| 0;
```

2 Overview

The intention of the MM9Z1x638 IBS is to measure voltage, current, and temperature of a battery system. The primary target application is to monitor 12 V (lead acid) starter or auxiliary batteries. Other battery monitoring applications like uninterruptible power supply (UPS), emergency/backup supplies (e.g. used in elevators) are also feasible.

Figure 1 shows an application block diagram of such a 12 V monitoring system.

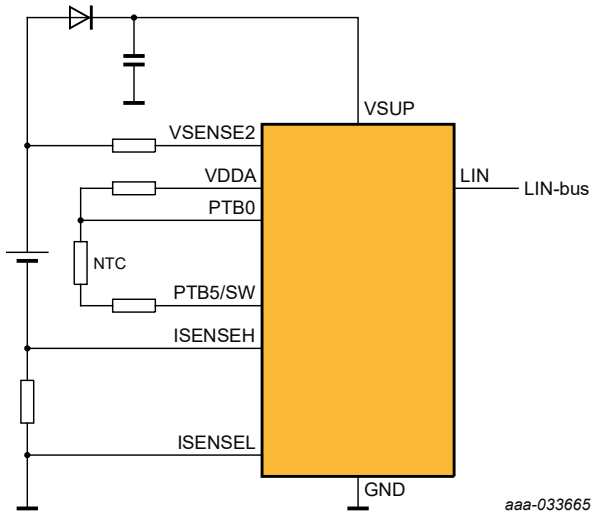

Figure 1. 12 V lead acid battery management system (BMS)

To ensure the required performance and accuracy in the application, perform a system calibration.

Section 3 explains in detail how the device measurement chain compensation features work.

Section 4 explains how NXP calibrates each device and what information NXP stores inside the IFR memory.

Section 5 gives guidelines on how to perform a system calibration.

Section 6.1 gives a PYTHON script for performing the calibration calculating from Section 5.

Section 6.2 shows the output of the PYTHON script from Section 6.1.

3 Acquisition chain compensation

3.1 Startup trimming

Some internal blocks, like oscillator and band gap references, require trimming to ensure proper operation of the MM9Z1x638. Perform this startup trimming after each power on (only needed after a power on).

To perform startup trimming, simply copy values from the information row (IFR) flash memory to specific trim registers. NXP measured the required values during final production [automated test equipment (ATE)] and stored those values in the IFR memory.

Table 1 summarizes the trim registers.

Table 1. Trim register overview

Device parameter	Register
Internal RC (IRC) oscillator on microcontroller	CPMUIRCTRIM[1]
Low-power oscillator	TRIM_OSC
Band gaps, voltage references, and low voltage threshold behavior	TRIM_BG0
	TRIM_BG1

[1] The device automatically trims the IRC oscillator (CPMUIRCTRIM register) at startup.

3.2 Compensation

The MM9Z1x638 has three dedicated sigma delta analog-to-digital converters (ADC), one for current measurements (ISENSE), one for voltage measurements (VSENSE) and one for temperature measurements (TSENSE).

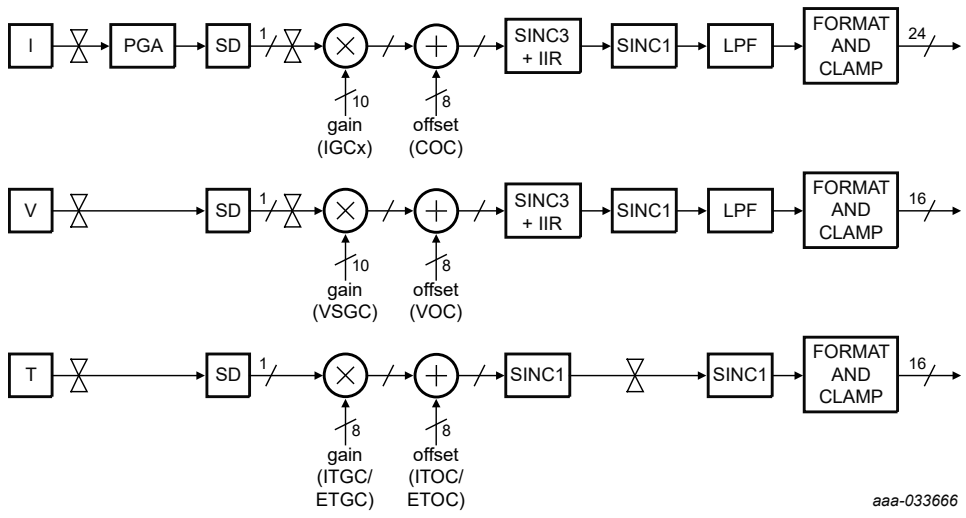

Figure 2. Acquisition chains

The principle structure of each acquisition chain is similar, but differs due to the specific requirements for each chain. For example, the ISENSE has a programmable gain amplifier (PGA) to amplify the tiny voltages provided by the shunt-based current measurement.

Each acquisition channel requires a compensation of device-to-device tolerances, as well as temperature effects. The acquisition channels utilize a linear compensation scheme: multiplying the sigma delta output with a gain compensation value and adding an offset compensation value.

Note: Adding the offset compensation value is the second operation (performed after the gain compensation). Any change of the gain compensation value also causes a scaling of the offset value, to maintain the offset compensation value magnitude.

Table 2 lists the details (register encoding/resolution, default value, and code) of the gain compensation and offset compensation values for each channel.

Table 2. Acquisition channel compensation values

Channel input		Offset compensation					Gain compensation				
		Unit	LSB	Default value	Default code	Format	Unit	LSB	Default value	Default code	Format
Voltage sense	VSENSE3	mV	3.998	0	00h	sint8	1	0.000488	0.9922	200h	10 bit with offset
	VSENSE2	mV	2.000	0	00h	sint8	1	0.000488	1.0679	200h	10 bit with offset
	VSENSE1	mV	1.000	0	00h	sint8	1	0.000488	1.2207	200h	10 bit with offset
	VSENSE0	mV	1.000	0	00h	sint8	1	0.000488	0.7632	200h	10 bit with offset
	PTB[4:0]	mV	0.100	0	00h	sint8	1	0.000488	0.7632	200h	10 bit with offset
Current sense	gain 4	mV	0.025602	0	00h	sint8	1	0.000977	1.4902	200h	10 bit with offset
	gain 16	mV	0.006400	0	00h	sint8	1	0.000977	1.4902	200h	10 bit with offset
	gain 64	mV	0.001600	0	00h	sint8	1	0.000977	1.4902	200h	10 bit with offset
	gain 256	mV	0.000400	0	00h	sint8	1	0.000977	1.4902	200h	10 bit with offset
Temperature sense	ETS PTB[4:0]	mV	0.152	0	00h	sint8	1	0.000977	1	80h	8 bit with offset
	ITS	K	0.06384	0	00h	sint8	1	0.000977	1	80h	8 bit with offset

Note: The resolution of the compensation registers (see least significant bit (LSB) Table 2) are different than the resolution of the corresponding acquisition channels (see Table 4 and Table 5).

Table 3 summarizes the available MM9Z1x638 compensation registers.

Table 3. Compensation register overview

Channel	Device parameter	Register
VSENSE	VSENSE gain for channel; three temperatures[1]	COMP_VSG_VSENSE, COMP_TVSG_VSENSE
	VSENSE offset for channel; three temperatures[1]	COMP_VO_VSENSE
ISENSE	PGA offset per gain; filled by auto zero sequence	COMP_PGAx
	ISENSE gain for each gain; three temperatures[1]	COMP_IG256, COMP_TIG256, COMP_IG64, COMP_TIG64, COMP_IG16, COMP_TIG16, COMP_IG4, COMP_TIG4
	ISENSE offset; four paged registers, using B_ACQ_GAIN for paging	COMP_IO(Gxxx)
TSENSE	internal temperature sensor (ITS) gain	COMP_ITG
	ITS offset	COMP_ITO
	external temperature sensor (ETS) gain	COMP_ETG
	ETS offset	COMP_ETO

[1] A code for room temperature and delta code for hot and cold temperatures.

Note: Perform a regular PGA offset canceling using an automated PGA auto zero routine. For further details, see MM9Z1x638 data sheet.

3.2.1 VSENSE compensation

This section details how the compensation for the VSENSE chain is operating. Calculate the result ACQ_VOLT of the VSENSE chain with Equation 1:

$$ACQ\_VOLT = V_{in} \times \left(DIV_{ch} \times \frac{2^{16} - 1}{V_{ref}} \times gain_{ch} \right) + offset_{ch} \quad (1)$$

With

ACQ_VOLT: VSENSE result register

V_{in}: voltage on the input

DIV_{ch}: divider for the channel

$gain_{ch}$: gain compensation value for the channel

$offset_{ch}$: offset compensation value for the channel

Equation 2 gives the nominal (for default $gain_{ch}$ and $offset_{ch}$) ADC resolution over the whole chain on the input.

$$resV_{in(ch)} = \frac{1}{DIV_{ch}} \times \frac{V_{ref}}{2^{16} - 1} \times \frac{1}{gain_{ch}} \quad (2)$$

Table 4 summarizes the various properties for the different VSENSE channels.

Table 4. VSENSE channel properties

Channel	Divider	Gain [nom]	Offset [nom] (mV)	$resV_{in}$ (mV/LSB)
VSENSE3	1/52	0.9922	0	1
VSENSE2	1/28	1.0679	0	0.5
VSENSE1	1/16	1.2207	0	0.25
VSENSE0	1/10	0.7632	0	0.25
VSENSE_EXT	1/1	0.7632	0	0.025

Note: Compensation is trying to achieve the nominal ADC resolution and to eliminate offset errors. Typically, this compensation requires gain and offset compensation values different to the nominal values.

Figure 3 depicts the VSENSE compensation principle and configuration options.

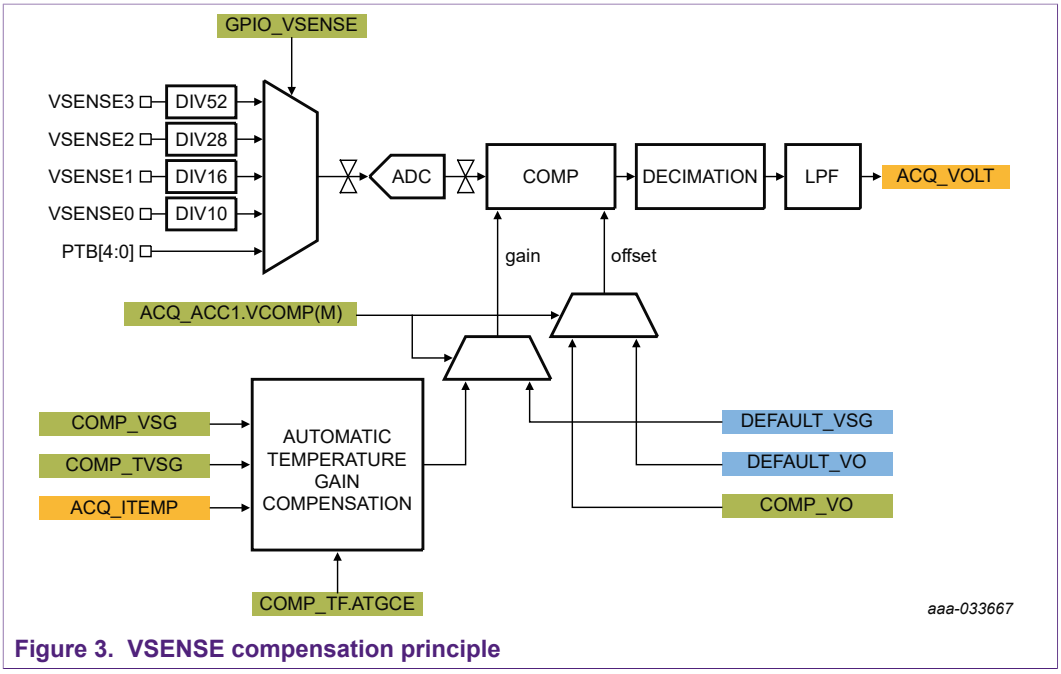

Figure 3. VSENSE compensation principle

VSENSE compensation features:

- Disable compensation (e.g. used for calibration measurements)
- Manual temperature compensation in software, e.g. using lookup tables
- Automatic temperature gain compensation (with static offset compensation)

It is sufficient to use a static (room temperature) value COMP_VO with VSENSE chopper mode enabled (ACQ_ACC1.CVCHOP(M) = 1). Otherwise, use a temperature compensation of the offset error with VSENSE chopper mode disabled (ACQ_ACC1.CVCHOP(M) = 0).

Use the ACQ_ACC1.VCOMP(M) bit to enable/disable the VSENSE channel compensation.

- If disabled, the device uses the default compensation code DEFAULT_VSG = 200h and DEFAULT_VO = 00h (see Table 2). Which results, e.g. for the VSENSE2, in a gain compensation value of 1.0679 and an offset compensation value of 0 mV.
- If enabled, the device uses the compensation values provided in the COMP_VSG, COMP_VO and COMP_TVSG (only for ATGCE = 1) registers.

To enable automatic gain temperature compensation (chip temperature), set the COMP_TF.ATGCE bit. Temperature adjustments only happen to the gain using the COMP_TVSG values. The offset value is constant over temperature. For more details, see Section 3.2.5.

It is also possible to update manually the COMP_VSG and COMP_VO registers to compensate temperature effects based on the measured chip temperature. In this case, the content of the register COMP_TVSG has no meaning. For more details, see Section 3.2.4.

Note: The COMP_TF.ATGCE bit impacts both VSENSE and ISENSE acquisition channels. The device does not generate calibration requests (interrupts) with ATGCE enabled. Choose to use either ATGCE or calibration requests.

3.2.2 ISENSE compensation

This section details how the compensation for the ISENSE chain is operating. Calculate the result ACQ_CURR of the ISENSE chain with Equation 3:

$$\mathrm{ACQ\_CURR} = \left[V_{in} \times \left(\mathrm{IGAIN} \times \frac{2^{16} - 1}{V_{ref}} \times \mathrm{gain}_{\mathrm{IGAIN}} \right) + \mathrm{offset}_{\mathrm{IGAIN}} \right] \times \frac{128}{\mathrm{IGAIN}} \quad (3)$$

With

ACQ_CURR: ISENSE result register

V_{in}: voltage on the input; $V_{in} = I \times R_{shunt}$

IGAIN: gain of the PGA (256, 64, 16 or 4)

$\mathrm{gain}_{\mathrm{IGAIN}}$: gain compensation value (one per IGAIN)

$\mathrm{offset}_{\mathrm{IGAIN}}$: offset compensation value (one per IGAIN)

$\frac{128}{\mathrm{IGAIN}}$: used to format the result to a fixed resolution for COMP_TF.IRSEL = 010b (100 µΩ).

Equation 4 gives the nominal (for default $\mathrm{gain}_{\mathrm{IGAIN}}$ and $\mathrm{offset}_{\mathrm{IGAIN}}$) ADC resolution for the whole chain.

$$\mathrm{resV}_{\mathrm{in(IGAIN)}} = \frac{1}{128} \times \frac{V_{ref}}{2^{16} - 1} \times \frac{1}{\mathrm{gain}_{\mathrm{IGAIN}}} \quad (4)$$

Table 5 summarizes the various properties for ISENSE chain, for the different PGA gains.

Table 5. ISENSE channel properties (COMP_TF.IRSEL = 010b)

Channel	PGA gain (IGAIN)	Gain [nom]	Offset [nom] (µV)	$resV_{in(IGAIN)}$ (µV/LSB)
ISENSE	256	1.4902	0.0	0.1
	64	1.4902	0.0	0.1
	16	1.4902	0.0	0.1
	4	1.4902	0.0	0.1

Table 5 shows that the $resV_{in(IGAIN)}$ is always 0.1 µV/LSB. For a 100 µΩ shunt resistor, this device feature results in a fixed resolution of $resI_{in}$ = 1 mA/LSB independent of the PGA gain.

Note: Compensation is trying to achieve the nominal ADC resolution and to eliminate offset errors. Typically, this compensation requires gain and offset compensation values different to the nominal ones.

Figure 4 depicts the ISENSE compensation principle and configuration options.

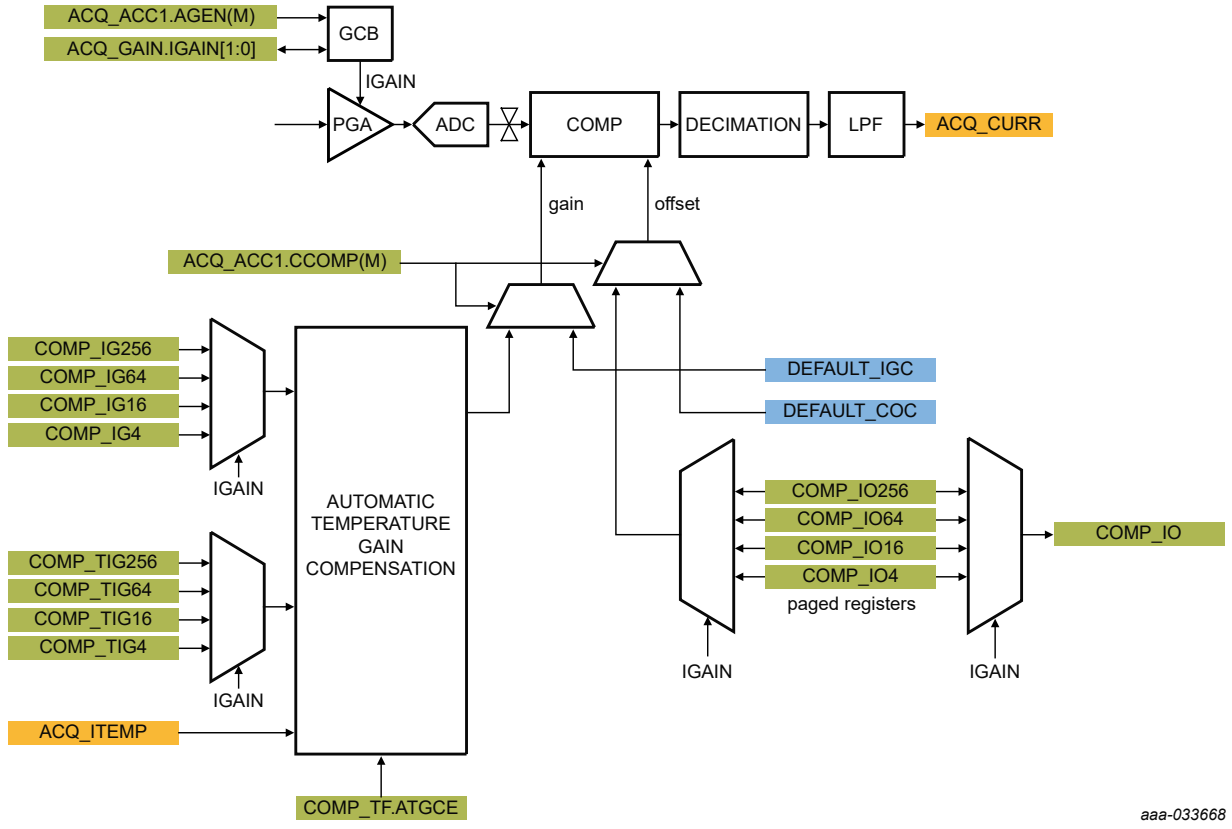

Figure 4. ISENSE compensation principle

ISENSE compensation features:

- Automatic or manual PGA gain selection
- Enable/disable of compensation (e.g. used for calibration measurements)
- Manual temperature compensation of gain and offset in software, e.g. using lookup tables.
- Automatic temperature compensation of gain (with static offset compensation)

As the ISENSE always incorporates a chopper mode, the use of one room-only COMP_IO value is sufficient.

Enable the compensation for the ISENSE channel using the ACQ_ACC1.CCOMP(M) bit.

- If disabled, the device uses the default compensation code DEFAULT_IGC = 200h and DEFAULT_COC = 00h (see Table 2). Which results in a gain compensation value of 1.4902 and an offset compensation value of 0.0 µV.
- If enabled, the device uses the compensation values provided in the COMP_IGxxx, COMP_IOxxx and COMP_TIGxxx (only for COMP_TF.ATGCE = 1) registers.

To enable automatic gain temperature compensation (chip temperature), set the COMP_TF.ATGCE bit. The device only adjusts the gain based on the COMP_TIGxxx values. The offset value is constant over temperature. For more details, see Section 3.2.5.

It is also possible to update manually the COMP_IGxxx and COMP_IOxxx registers to compensate temperature effects based on the measured chip temperature. In this case, the content of the registers COMP_TIGxxx has no meaning. For more details, see Section 3.2.4.

Note: The COMP_TF.ATGCE bit impacts both VSENSE and ISENSE acquisition channels. The device does not generate calibration requests (interrupts) with ATGCE enabled. Choose to use either ATGCE or calibration requests.

3.2.3 TSENSE compensation

The TSENSE acquisition channel handles both, internal chip temperature sensing as well as sensing external temperatures using negative temperature coefficient (NTC) resistor.

Separate compensation registers are available for ITS and ETS.

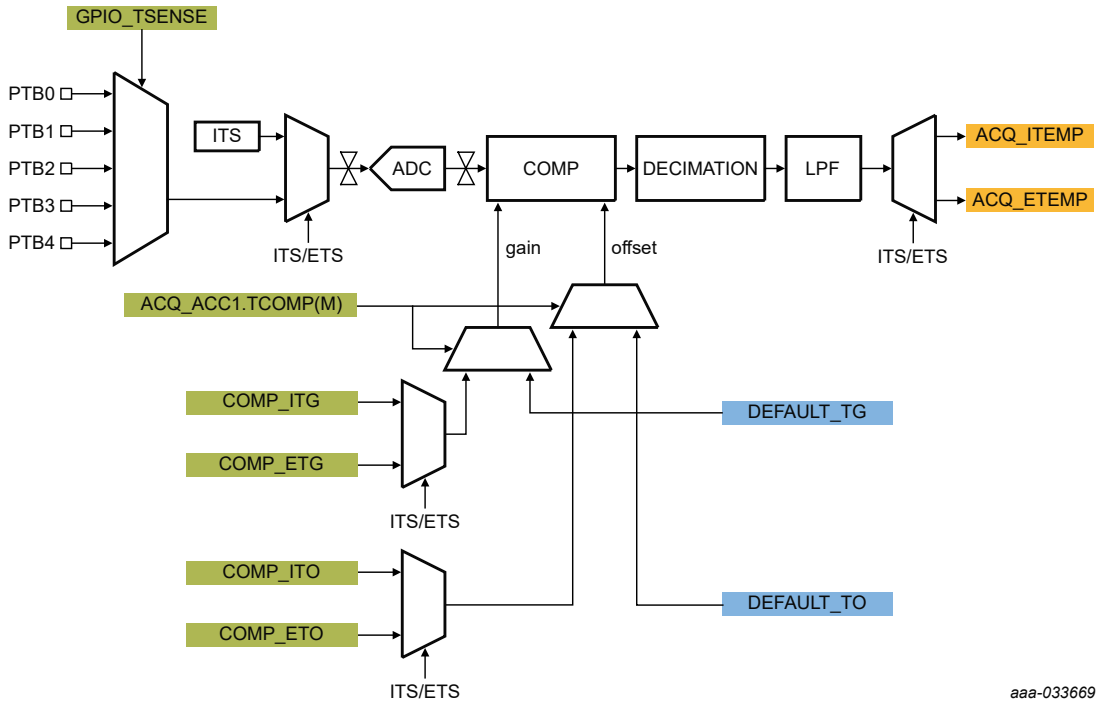

Figure 5. TSENSE compensation principle

3.2.3.1 TSENSE chip temperature (ITS)

Calculate the result ACQ_ITEMP of the TSENSE chain with Equation 5:

$$ACQ\_ITEMP = T_{chip} \times \left(\frac{2^{16} - 1}{524} \times gain + offset\right) \quad (5)$$

With

ACQ_ITEMP: chip temperature result register

T_{chip}: chip temperature in kelvin

gain: gain compensation value for ITS

offset: offset compensation value for ITS

Equation 6 gives the nominal (for default gain and offset) ADC resolution.

$$resT_{chip} = \frac{524}{2^{16} - 1} = 0.008 \quad (6)$$

Table 6 summarizes the properties for TSENSE chain:

Table 6. TSENSE (ITS) channel properties

Channel	Gain [nom]	Offset [nom] (K)	$resT_{chip}$ (K/LSB)
TSENSE (ITS)	1.0	0	0.008

Note: Compensation is trying to achieve the nominal ADC resolution and to eliminate offset errors. Typically, this compensation requires gain and offset compensation values different to the nominal ones.

3.2.3.2 TSENSE external (ETS)

The TSENSE external measurements are ratiometric (to V_{DDA}) measurements, accurately measuring ratios, e.g. the ratio of a resistive divider. TSENSE external is not suited to measure absolute voltages as the accuracy of the result depends on the accuracy of the V_{DDA} supply.

Calculate the result ACQ_ETEMP of the TSENSE chain with Equation 7:

$$ACQ\_ETEMP = (V_{PTBx} - V_{PTB5}) \times \left(\frac{2^{16} - 1}{0.5 \times (V_{DDA} - V_{PTB5})} \times gain + offset\right) \quad (7)$$

With

ACQ_ETEMP: external TSENSE result register

V_{PTBx}: voltage on the input PTBx

V_{PTB5}: voltage on the ground switch PTB5

V_{DDA}: analog supply voltage (typ. 2.5 V)

gain: gain compensation value for ETS

offset: offset compensation value for ETS

Equation 8 gives the nominal (for default gain and offset) ADC resolution.

$$resV_{temp} = \frac{0.5 \times (V_{DDA} - V_{PTB5})}{2^{16} - 1} \quad (8)$$

Table 7 summarizes the properties for TSENSE chain:

Table 7. TSENSE (ETS) channel properties

Channel	Gain [nom]	Offset [nom] (mV)	$resV_{temp}$ (mV)
TSENSE (ETS)	1.0	0	0.019

Note: Compensation is trying to achieve the nominal ADC resolution and to eliminate offset errors. Typically, this compensation requires gain and offset compensation values different to the nominal ones.

TSENSE compensation features:

- Disable compensation (e.g. used for calibration measurements)

Enable/disable the compensation for the TSENSE channel using the ACQ_ACC1.TCOMP(M) bit.

- If disabled, the default compensation code DEFAULT_TG = 80h and DEFAULT_TO = 00h (see Table 2) is valid. Resulting in a gain compensation value of 1.0 and an offset compensation value of 0 K respectively 0 mV.
- If enabled, the compensation values provided in the COMP_ITG, COMP_ITO respectively COMP_ETG, COMP_ETO registers are valid.

Note: Properly compensation of the ITS channel is mandatory for correct operation of the VSENSE and ISENSE temperature compensation. It is mandatory to enable and regularly sample the chip temperature (ITS).

3.2.4 Manual temperature compensation

Using the internal temperature sensor enables compensation of chip temperature depending on effects. Obviously, it is possible to poll regularly the chip temperature information and adjust based on it. But the MM9Z1x638 device offers a more sophisticated method.

This method generates recalibration requests (interrupts) based on a preset temperature threshold. This mechanism is also available to wake up from low-power modes as, e.g., required for accurate coulomb counting. As reaction to the recalibration request, adopt all desired compensation values and update new temperature thresholds in the COMP_TMAX and COMP_TMIN registers.

Figure 6 depicts the generation of the recalibration requests.

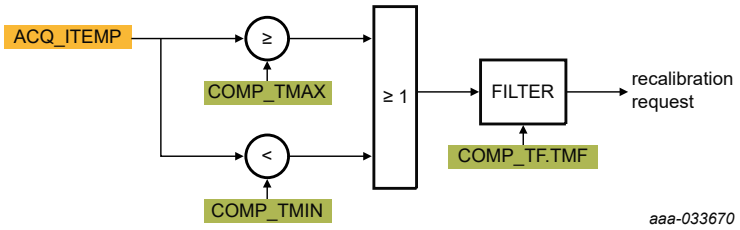

Figure 6. Recalibration requests

Table 8. Registers for recalibration requests

Register	Usage
COMP_TMAX.TCMAX[7:0]	upper temperature threshold for recalibration (only the high-order byte)
COMP_TMIN.TCMIN[7:0]	lower temperature threshold for recalibration (only the high-order byte)
COMP_TF.TMF[2:0]	counter threshold value A up/down counter increments with each ACQ_ITEMP result above the T_{max} or below the T_{min} thresholds. Otherwise (if within T_{max}/T_{min} range) it decrements down until zero. Exceeding the configured counter threshold value triggers a recalibration request.
COMP_CTL.CALIE(M)	recalibration interrupt enable
COMP_SR.CALF	recalibration request status flag

Note: The device does not generate calibration requests (interrupts) with ATGCE enabled. Choose using either ATGCE or calibration requests.

Manual temperature (gain and offset) compensation is the recommended method, for VSENSE channel chopper off mode (ACQ_ACC1.CVCHOP(M) = 0). Clear COMP_TF.ATGCE to use manual temperature compensation. Use a gain and an offset temperature compensation to achieve the specified accuracy.

For VSENSE channel chopper on mode (ACQ_ACC1.CVCHOP(M) = 1), the recommended method is the auto temperature gain compensation.

3.2.5 Auto temperature gain compensation

The MM9Z1x638 device provides an automatic temperature compensation mechanism, which adjusts the used gain compensation value in dependency of the chip temperature measurement result (register ACQ_ITEMP).

The following explanation uses the VSENSE channel as example, but same applies to the ISENSE channel.

Note: To ensure correct operation of this feature, regularly acquire the internal chip temperature.

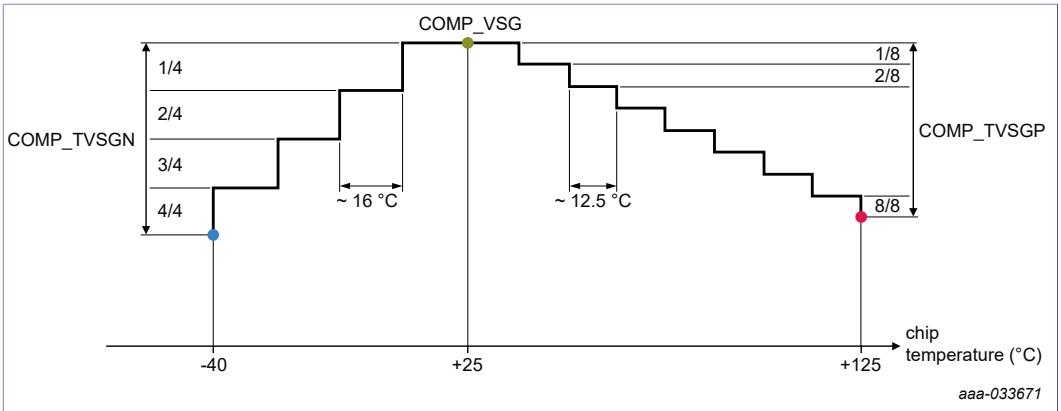

Figure 7. Principle of gain compensation (VSENSE)

Figure 7 depicts the principle of the gain compensation adjustment.

The adjustment divides the overall temperature range in two sections:

1. Section room temperature to hot temperature
   This section uses a linear interpolation, in eight steps of approximately 12.5 °C, between COMP_VSG and COMP_VSG + COMP_TVSGP. Resulting in a compensation value of COMP_VSG + 8/8 × COMP_TVSGP at 125 °C.
2. Section room temperature to cold temperature
   This section uses a linear interpolation, in four steps of approximately 16 °C, between COMP_VSG and COMP_VSG + COMP_TVSGN. Resulting in a compensation value of COMP_VSG + 4/4 × COMP_TVSGN at −40 °C.

The automatic adjustment uses the same hardware blocks as used for the recalibration requests; see Figure 6. The automatic adjustment is using the overlapping temperature bands shown in Table 9 and Table 10 resulting in a hysteresis like step function.

Table 9. Temperature bands (room temperature to hot temperature)

Band	T_{min} (°C)	T_{max} (°C)	TCMIN	TCMAX
+9/8	126.2	249.1	C3h	FFh
+8/8	113.9	138.5	BDh	C9h
+7/8	101.6	126.2	B7h	C3h
+6/8	89.3	113.9	B1h	BDh
+5/8	75.0	101.6	AAh	B7h
+4/8	62.7	89.3	A4h	B1h
+3/8	50.4	75.0	9Eh	AAh
+2/8	38.1	62.7	98h	A4h
+1/8	25.9	50.4	92h	9Eh
Room	9.5	38.1	8Ah	98h

Table 10. Temperature bands (room temperature to cold temperature)

Band	T_{min} (°C)	T_{max} (°C)	TCMIN	TCMAX
+5/5	−273.2	−37.6	00h	73h
+4/4	−54.0	−23.3	6Bh	7Ah
+3/4	−37.6	−6.9	73h	82h
+2/4	−23.3	+9.5	7Ah	8Ah
+1/4	−6.9	+25.9	82h	92h
Room	+9.5	+38.1	8Ah	98h

For example, let us assume the COMP_TMIN.TCMIN and COMP_TMAX.TCMAX values loaded with the room band values 8Ah and 98h. Resulting in a lower threshold of 9.5 °C and an upper threshold of 38.1 °C.

If the temperature increases over 38.1 °C, the device selects the next upper temperature band. This temperature band change adopts the compensation code by 1/8th. The new temperature band uses lower/upper thresholds of 25.9 °C/50.4 °C. If the temperature increases further over 50.4 °C, the device selects the next upper band. If instead the temperature decreases under 25.9 °C, the device selects the room band.

See Table 12 for the VSENSE compensation values and Table 13 for the ISENSE compensation values stored in the IFR memory.

Automatic temperature gain compensation is the recommended method for VSENSE channel chopper on mode (ACQ_ACC1.CVCHOP(M) = 1). Set COMP_TF.ATGCE to use automatic temperature gain compensation. This method uses a gain temperature compensation with a fixed offset compensation value to achieve the specified accuracy.

For VSENSE channel chopper off mode (ACQ_ACC1.CVCHOP(M) = 0) the recommended method is the manual temperature gain compensation.

4 Device calibration

4.1 Overview

During final test (using ATE), NXP individually calibrates each single MM9Z1x638 device and stores respective information in the IFR memory of the device. The required data collection happens at three temperatures (cold, room, and hot).

The device calibration therefore provides temperature characteristics, enabling the user to perform later a system calibration at room temperature only.

NXP uses the following calibration points during device calibration:

Table 11. Device calibration points (used on ATE)

Channel	Points					Unit
VSENSE3	6.5	9.4	21.5	33.5	50	V
VSENSE2	3.5	5	12	18	28	V
VSENSE1	2	2.8	6.5	10.3	16	V
VSENSE0	1.25	1.8	4.1	6.5	10	V
VSENSE_EXT	0.13	0.18	0.64	0.8	0.95	V
ISENSE (gain 4)	−150	−100	+50	+100	+150	mV
ISENSE (gain 16)	−60	−40	+20	+40	+60	mV
ISENSE (gain 64)	−15	−10	+8	+10	+15	mV
ISENSE (gain 256)	−4	−3.5	+3	+3.5	+4	mV
TSENSE ETS	0.125	0.25	0.5	0.75	1.0	V
TSENSE ITS	+85	+25	−40	-	-	°C

The provided compensation information (in the IFR memory) differs depending on the needs of each channel.

4.2 VSENSE compensation values

Table 12 lists the device compensation information for the VSENSE chain.

Table 12. VSENSE compensation data (IFR)

Channel	Compensation values	Coding	Comment
VSENSE0	COMP_VSG_VSENSE0	10 bit	gain compensation code at 25 °C
	COMP_TVSGCP_VSENSE0	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TVSGCN_VSENSE0	5 bit (signed)	gain compensation delta-code at −40 °C
	COMP_VO_VSENSE0	8 bit	offset compensation code at 25 °C
	COMP_TVSOCP_VSENSE0	5 bit (signed)	offset compensation delta-code at 125 °C
	COMP_TVSOCN_VSENSE0	5 bit (signed)	offset compensation delta-code at −40 °C
VSENSE1	COMP_VSG_VSENSE1	10 bit	gain compensation code at 25 °C
	COMP_TVSGCP_VSENSE1	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TVSGCN_VSENSE1	5 bit (signed)	gain compensation delta-code at −40 °C
	COMP_VO_VSENSE1	8 bit	offset compensation code at 25 °C
	COMP_TVSOCP_VSENSE1	5 bit (signed)	offset compensation delta-code at 125 °C
	COMP_TVSOCN_VSENSE1	5 bit (signed)	offset compensation delta-code at −40 °C

Channel	Compensation values	Coding	Comment
VSENSE2	COMP_VSG_VSENSE2	10 bit	gain compensation code at 25 °C
	COMP_TVSGCP_VSENSE2	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TVSGCN_VSENSE2	5 bit (signed)	gain compensation delta-code at −40 °C
	COMP_VO_VSENSE2	8 bit	offset compensation code at 25 °C
	COMP_TVSOCP_VSENSE2	5 bit (signed)	offset compensation delta-code at 125 °C
	COMP_TVSOCN_VSENSE2	5 bit (signed)	offset compensation delta-code at −40 °C
VSENSE3	COMP_VSG_VSENSE3	10 bit	gain compensation code at 25 °C
	COMP_TVSGCP_VSENSE3	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TVSGCN_VSENSE3	5 bit (signed)	gain compensation delta-code at −40 °C
	COMP_VO_VSENSE3	8 bit	offset compensation code at 25 °C
	COMP_TVSOCP_VSENSE3	5 bit (signed)	offset compensation delta-code at 125 °C
	COMP_TVSOCN_VSENSE3	5 bit (signed)	offset compensation delta-code at −40 °C
VSENSE_EXT	COMP_VSG_VSENSE_EXT	10 bit	gain compensation code at 25 °C
	COMP_TVSGCP_VSENSE_EXT	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TVSGCN_VSENSE_EXT	5 bit (signed)	gain compensation delta-code at −40 °C
	COMP_VO_VSENSE_EXT	8 bit	offset compensation code at 25 °C
	COMP_TVSOCP_VSENSE_EXT	5 bit (signed)	offset compensation delta-code at 125 °C
	COMP_TVSOCN_VSENSE_EXT	5 bit (signed)	offset compensation delta-code at −40 °C

Note: NXP performs the VSENSE device calibration in chopper off mode (ACQ_ACC1.CVCHOP(M) = 0) to obtain the temperature characteristics of the offset error. Knowing the temperature behavior enables offset temperature compensation [COMP_VO_CHOPOFF = f(T_{chip})].

If using chopper mode on (ACQ_ACC1.CVCHOP(M) = 1), the application does not require an offset temperature compensation. In this case, apply the system calibration obtained (room only) offset compensation value.

4.3 ISENSE compensation values

Table 13 lists the device compensation information for the ISENSE chain.

Table 13. ISENSE compensation data (IFR)

PGA gain	Compensation values	Coding	Comment
256	COMP_IG256	10 bit	gain compensation code at 25 °C
	COMP_TIG256P	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TIG256N	5 bit (signed)	gain compensation delta-code at −40 °C
64	COMP_IG64	10 bit	gain compensation code at 25 °C
	COMP_TIG64P	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TIG64N	5 bit (signed)	gain compensation delta-code at −40 °C

PGA gain	Compensation values	Coding	Comment
16	COMP_IG16	10 bit	gain compensation code at 25 °C
	COMP_TIG16P	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TIG16N	5 bit (signed)	gain compensation delta-code at −40 °C
4	COMP_IG4	10 bit	gain compensation code at 25 °C
	COMP_TIG4P	5 bit (signed)	gain compensation delta-code at 125 °C
	COMP_TIG4N	5 bit (signed)	gain compensation delta-code at −40 °C

Note: The IFR memory only provides a gain compensation value and no offset compensation value. The application does not require an offset temperature compensation as the ISENSE channel always operates in chopper mode on. Apply the system calibration obtained (room only) offset compensation value.

4.4 TSENSE compensation values

Table 14 lists the device compensation information for the TSENSE chain.

Table 14. TSENSE compensation data (IFR)

Channel	Compensation values	Coding	Comment
ITS	COMP_ITG	8 bit	gain compensation code
	COMP_ITO	8 bit (signed)	offset compensation code
ETS	COMP_ETG_ROOM	8 bit	gain compensation code at 25 °C
	COMP_ETG_HOT	8 bit	gain compensation code at 125 °C
	COMP_ETG_COLD	8 bit	gain compensation code at −40 °C

For proper operation (correct T_{chip} evaluation) it is mandatory to apply the ITS compensation values.

The IFR memory only provides gain compensation values for the ETS.

5 System calibration

This section explains the handling of a system calibration and gives guidelines.

5.1 Overview

A system calibration happens on system level (on module/PCB level). It addresses effects/errors introduced through device (acquisition channel) configuration, soldering and external components (depending on use case).

The principle of the calibration is to remove errors such, that the real behavior is as close to the desired (ideal) behavior as possible. The method to achieve this calibration, is to apply compensation values. Comparing ideal and real behavior provides the compensation values.

Typically, a calibrated high precision measurement equipment, e.g. a calibrated high precision digital multimeter (DMM), provides the ideal behavior data points.

For our use case, it is sufficient to use one gain compensation and one offset compensation value (linear system).

NXP calibrated each device at final testing; see Section 4. Apply the device compensation values when performing the incremental system calibration.

The basic required steps are (see Figure 8):

- Characterize system behavior (applying device compensation values)
- Estimate deviation between real and ideal behavior (calculate delta between ideal behavior and measured behavior)
- Calculate incremental compensation values (delta values)
- Apply compensation during operation and verify system behavior

Utilize the device EEPROM or FLASH for storing the incremental compensation values inside the device.

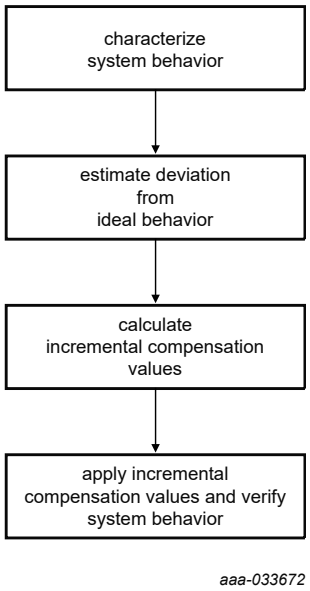

Figure 8. System calibration flow diagram

5.2 Characterize system behavior

To characterize the system behavior, execute calibration measurements and compare the results against exact measurements. Use precision measurement equipment to obtain exact measurement results.

For a good estimation of the acquisition channels behavior, e.g. a 5-point measurement is suitable. The selection of the individual data points should consider application requirements like:

- Desired input range
- Area where highest accuracy is desired
- Capability of the module/device (e.g. minimum supply voltage for V_{sup})
- Capability of equipment (e.g. max. I_{out} of power supplies, max. input range of DMM and related accuracy in the range)
- Select current points for ISENSE calibration such that no gain switching occurs

Table 15 provides example data points (5-point calibration).

Table 15. Example calibration points

Channel	Points					Unit
VSENSE3	9.0	12.0	15.0	25.0	35.0	V
VSENSE2	5.0	8.0	12.0	16.0	20.0	V
VSENSE1	4.0	6.0	8.0	10.0	16.0	V
VSENSE0	1.0	2.0	3.0	4.0	9.0	V
VSENSE_EXT	0.1	0.25	0.5	0.75	0.95	V
ISENSE	−38.0	−19.0	+10.0	+19.0	+38.0	A

The following subchapters give examples for a VSENSE and an ISENSE channel system calibration intended as guideline for setting up an own calibration setup.

5.2.1 VSENSE system calibration

Figure 9 shows an example of a setup for a VSENSE2 calibration.

The voltage supply supplies both the module as well as the calibration voltage for VSENSE2. In this example case, it is not possible to separate the module supply from the parameter to calibrate.

In general, it is advisable to separate supply of the module from the parameters of interest to avoid any crosstalk/disturbances.

A calibrated high precision DMM measures the real applied voltage between the BAT+ and GND terminals.

The device sends the measurement values to a PC-based GUI using the local interconnect network bus (LIN-bus).

Note: Take special care that DMM is measuring the right voltage. For example, take care for voltage drops caused by the modules supply current.

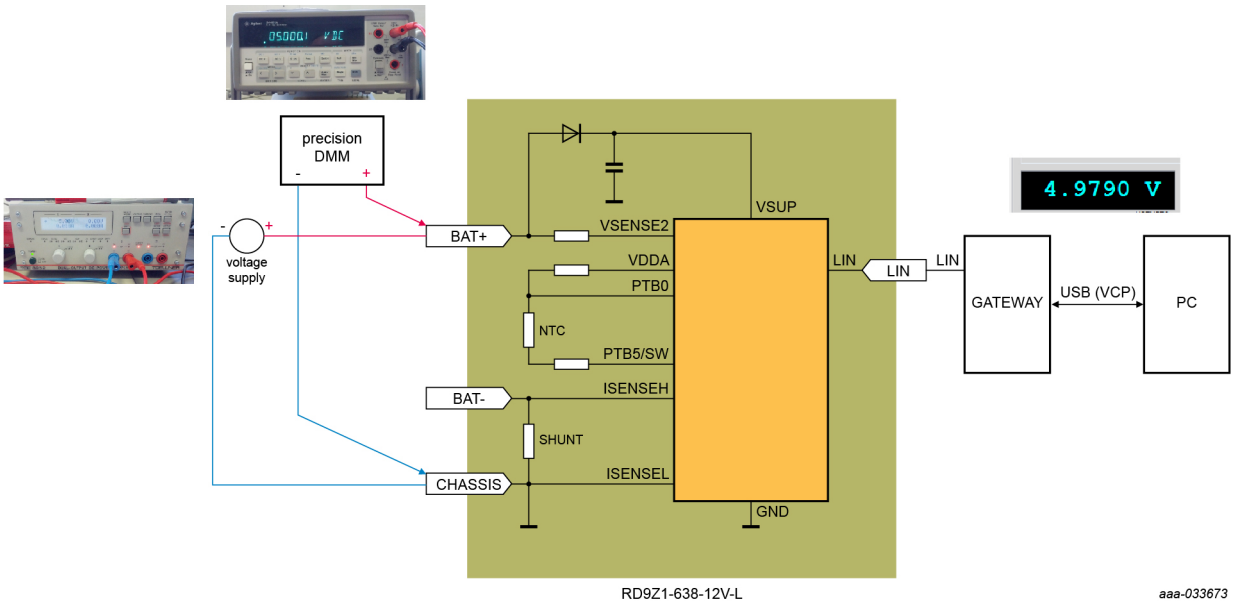

Figure 9. Example VSENSE2 calibration setup

The calibration measurements require special settings for the VSENSE channel. These settings typically differ from the settings applied in the application.

Different are for example:

- Using device compensation values instead of system calibration values
- Only activating VSENSE instead of all channels (VSENSE, ISENSE, and TSENSE)
- Polling of acquisition results instead of interrupt driven (CVMI interrupt)

Table 16 summarizes the ADC settings used for the VSENSE calibration measurements.

Table 16. Setup for VSENSEx calibration measurements

Configuration	Setting	Comment
Enable acquisition (ACQ) channels	VSENSE	ISENSE and TSENSE are disabled
Channel multiplexer (GPIO_VSENSE)	CH_VSENSEx	selected channel
Chopper	on	apply same setting as in application
Decimation/output data rate (ODR)	512 (1 kHz)	apply same setting as in application
Infinite impulse response (IIR) filter	1/32	apply same setting as in application
Low-pass filter	off	apply same setting as in application
Compensation	on	device compensation active
Compensation values	device	use device compensation values (IFR)
Auto temperature gain compensation (ATGCE)	off	off to avoid any temperature effects

Note: The ADC setup should use the same configuration as used in the application. Perform the calibration at ~25 °C.

To get a more reliable and noise reduced result, use the average over multiple measurements, e.g. average over 1000 samples.

5.2.2 ISENSE system calibration

Figure 10 shows an example of a setup for an ISENSE calibration.

One voltage supply supplies the module. A second supply provides the current for the current measurement path. An external calibration shunt provides exact current readings.

A calibrated high precision DMM measures the real applied current: voltage drop on calibration shunt.

The device sends the measurement values to a PC-based GUI using the LIN-bus.

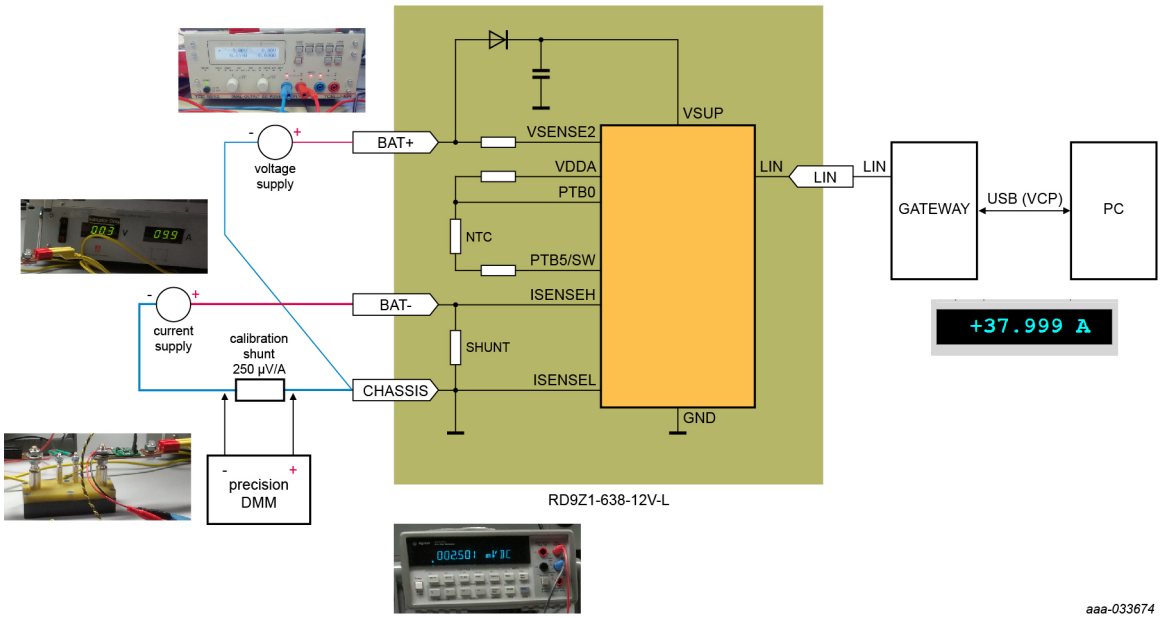

Figure 10. Example ISENSE calibration setup

The calibration requires special settings for the ISENSE channel. These settings typically differ from the settings applied in the application.

Different are for example:

- Using device compensation values instead of system calibration values
- Only activating ISENSE instead of all channels (VSENSE, ISENSE, and TSENSE) active
- Polling of acquisition results instead of interrupt driven (CVMI interrupt)

Table 17 summarizes the ADC settings used for the ISENSE calibration measurements.

Table 17. Setup for ISENSEx calibration measurements

Configuration	Setting	Comment
Enable ACQ channels	ISENSE	VSENSE and TSENSE are disabled
PGA auto zero	performed	to avoid saturation effects
Chopper	on	fixed on for ISENSE
Decimation/ODR	512 (1 kHz)	apply same setting as in application
IIR	1/32	apply same setting as in application
Shunt selection (COMP_TF.IRSEL)	100 µΩ	apply same setting as in application
PGA automatic gain control (AGC)	off (gain 256)	use only one gain
PGA switching thresholds (GCB)	-	AGC is off
Compensation [ACQ_ACC1.ICOMP(M)]	on	device compensation active
Compensation values	device	use device compensation values (IFR)
Auto temperature gain compensation (ATGCE)	off	off to avoid any temperature effects
Low-pass filter	off	apply same setting than in application

Note: The ADC setup should use the same configuration as used in the application. Perform the calibration at ~25 °C.

To get a more reliable and noise reduced result, use the average over multiple measurements, e.g. average over 1000 samples.

Only use one fixed PGA gain during ISENSE calibration. Apply the obtained incremental value to all four values (one per PGA gain) in the application.

5.3 Estimate deviation from ideal behavior

The calibration measurements deliver a vector of applied real values linked to the module measured values; see Figure 11.

In a first step, estimate the gain and offset errors using linear estimation; see Section 5.3.1. These values quantify the deviation from ideal behavior.

In a second step, use the gain and offset errors together with device-to-device specific values (IFR) and acquisition channel characteristics to calculate the required updates. Apply the required updates as incremental updates (gain steps and offset steps) to the used device compensation code (IFR values).

Figure 11 shows the overall data flow from the actual data collection (calibration measurements) to the resulting incremental compensation values for gain compensation and offset compensation.

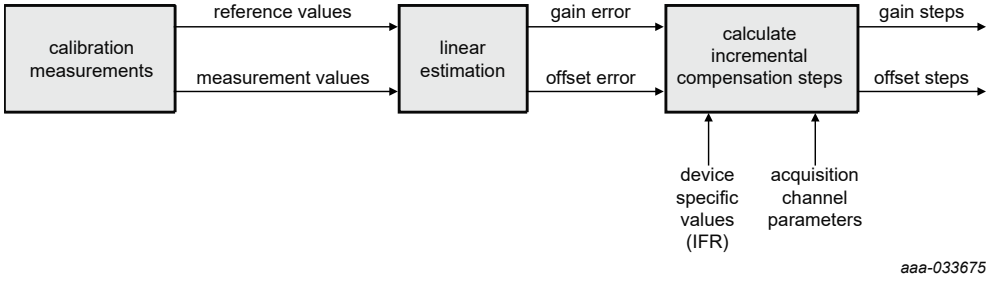

Figure 11. System calibration data flow

5.3.1 Calculating gain and offset errors using linear estimation

Linear estimation uses the least squares method to calculate a straight line through the known (x, y) samples.

Here the Xs (x_n) are the reference values (obtained with precision DMM) and the Ys (y_n) are the device measurement values.

Equation 9 describes the line.

$$y = mx + b \tag{9}$$

Calculate the slope m with

$$m = \frac{\sum_n (x_n - \bar{x})(y_n - \bar{y})}{\sum_n (x_n - \bar{x})^2} \quad (10)$$

and calculate the intercept b with

$$b = \bar{y} - m\bar{x} \quad (11)$$

With

n: index of the calibration point

x_n, y_n: values of the calibration measurement

$\bar{x}$, $\bar{y}$: average value over all n calibration measurements

Calculate the average value of $\bar{x}$ with

$$\bar{x} = \frac{1}{n}\sum_n x_n \quad (12)$$

Calculate the average value of $\bar{y}$ with

$$\bar{y} = \frac{1}{n}\sum_n y_n \quad (13)$$

Calculate the gain error and offset error, compared to an ideal behavior (m = 1 and b = 0), with

$$\text{gain error} = 1 - m \quad (14)$$

$$\text{offset error} = -b \quad (15)$$

5.4 Calculate incremental compensation code (gain steps and offset steps)

Scale the gain error and offset error evaluated in the previous chapter relative to the gain values and offset values used during the calibration measurements. Then encode the scaled values into the required device representation; see Table 2.

This method allows incremental adoption of the (for the system calibration) used compensation code to achieve compensated state.

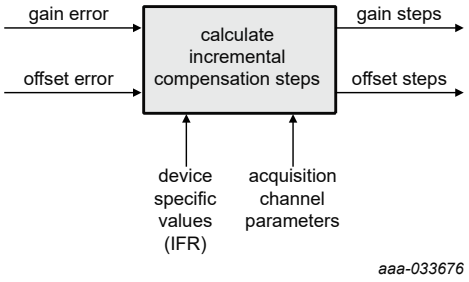

Figure 12. Calculation of incremental compensation steps

Use the following data to perform the calculation:

- Magnitude to compensate for (gain error and offset error)
- Gain used during calibration measurements (IFR data)
- Acquisition channel parameters (coding of IFR data); see Table 2

5.4.1 Calculate required gain steps

Decode the corresponding gain factor for a gain compensation code with the following formula:

$$\text{gain} = \text{gain}_{\text{def}} + \text{gain}_{\text{LSB}} \times \left(code - gain_{code(def)} \right) \quad (16)$$

With

gain: gain factor for a given code

code: actual code to decode

gain_{def}: gain compensation default value (from Table 2)

gain_{LSB}: the resolution of the gain compensation register (from Table 2)

$\text{gain}_{\text{code(def)}}$: gain compensation default code (from Table 2)

As the system calibration used the device compensation code, calculate the required gain adjustment Δgain with:

$$\Delta\text{gain} = \text{gain}_{\text{error}} \times \text{gain}_{\text{calib}} \quad (17)$$

With

$\text{gain}_{\text{error}}$: the gain error (evaluated in Section 5.3.1)

$\text{gain}_{\text{calib}}$: the gain factor used during system calibration (using Equation 16)

Calculate the gain steps by scaling the adjustment Δgain against the gain compensation register resolution and rounding the result to the closest integer.

$$\text{gain steps} = \text{round}\left(\frac{\Delta\text{gain}}{\text{gain}_{\text{LSB}}}\right) \tag{18}$$

With

Δgain: the gain adjustment (using Equation 17)

gain_{LSB}: the resolution of the gain compensation register (from Table 2)

5.4.2 Calculate required offset steps

As the offset compensation is performed following the gain compensation (see Figure 2), it is required to re-scale the measured offset to the adjusted gain factor (gainnew).

$$\Delta\text{offset} = \text{offset}_{\text{error}} \times \frac{\text{gain}_{\text{calib}}}{\text{gain}_{\text{new}}} \tag{19}$$

With

$\text{offset}_{\text{error}}$: the offset error (evaluated in Section 5.3.1)

$\text{gain}_{\text{calib}}$: the gain factor used during system calibration (using Equation 16)

gain_{new}: the gain factor after applying the new gain (gain code + gain steps from Equation 18 using Equation 16)

The offset steps can be calculated by scaling the Δoffset against the gain compensation register resolution and rounding the result to the closest integer.

$$\text{offset steps} = \text{round}\left(\frac{\Delta\text{offset}}{\text{offset}_{\text{LSB}}}\right) \tag{20}$$

5.5 Verify system calibration results

As noted before is the system calibration an incremental calibration to the device compensation code. Therefore, the total compensation code is the sum of the device compensation code and system compensation incremental steps.

Example code indicating how to combine device (IFR) and system calibration value (gain steps and offset steps):

```
/*
* Copyright 2016 - 2019 NXP, All rights reserved.
*
* SPDX-License-Identifier: BSD-3-Clause
*/

#define IFR_COMP_VO_VSENSE2 ((*(u8 *) 0x1FC0D8))
#define IFR_COMP_TVSO_VSENSE2 (*(u8 *) 0x1FC0FC)
#define IFR_COMP_TVSOP_VSENSE2 ((IFR_COMP_TVSO_VSENSE2>>4) & 0x0F)
#define IFR_COMP_TVSON_VSENSE2 ((IFR_COMP_TVSO_VSENSE2>>0) & 0x0F)

case CH_VSENSE2:
 B_GPIO_VSENSE = VSENSE2;
 B_COMP_VO   = IFR_COMP_VO_VSENSE2  + CalibValues.vsense[2].offsetsteps;
 B_COMP_VSG  = IFR_COMP_VSG_VSENSE2 + CalibValues.vsense[2].gainsteps;
 B_COMP_TVSG = IFR_COMP_TVSG_VSENSE2;
 break;
```

For verification, apply the gain step and the offset step. Perform again a multipoint measurement (likewise the system calibration measurements). Now the gain error and offset error should be below (or close to) the resolution of the respective gain and offset compensation resolution. For example, the resulting VSENSE2 gain error should be below (or close to) 0.000488.

5.6 Example (VSENSE2)

The following example shall demonstrate how the whole process of the system calibration works.

The system calibration measurements (five points) delivered these results

- reference_values [V]: 4.9994, 7.9999, 11.999, 15.999, 19.999
- measured_values [V]: 4.966, 7.9645, 11.9595, 15.956, 19.952

The device compensation values (from IFR) used during system calibration are

- IFR_COMP_VSG_VSENSE2: 518 (206h)
- IFR_COMP_VO_VSENSE2: −25 (E7h)

The gain error and offset error calculation results are

$gain_{error}$ [1] = 0.000918
$offset_{error}$ [V] = 0.028459

The required gain steps are

$gain_{calib}$ [1] = 1.0679 + 0.000488 × (518 − 512) = 1.070828
Δgain [1] = 0.000983
gain steps = round(0.000983 / 0.000488) = round(2.014)
gain steps = 2

The required offset steps are

$gain_{calib}$ [1] = 1.0679 + 0.000488 × (518 − 512) = 1.070828
$gain_{new}$ [1] = 1.0679 + 0.000488 × (518 + 2 − 512) = 1.071804
Δoffset = 0.028459 × 1.070828 / 1.071804 = 0.028433
offset steps = round(0.028433 / 0.002) = round(14.217)
offset steps = 14

5.7 Example (ISENSE)

The following example shall demonstrate how the whole process of the system calibration works.

The system calibration measurements (five points) delivered these results

- reference_values [A]: −38.000, −19.028, +10.052, +19.184, +38.140
- measured_values [A]: −37.554, −18.798, +9.958, +18.975, +37.714

The device compensation values (from IFR) used during system calibration are

- IFR_COMP_IG256: 522 (20Ah)
- IFR_COMP_IO_256: 0 (no value provided)

The gain error and offset error calculation results are

$gain_{error}$ [1] = 0.011438
$offset_{error}$ [A] = −0.013072

The required gain steps are

$gain_{calib}$ [1] = 1.4902 + 0.000977 × (522 − 512) = 1.49997
Δgain [1] = 0.017157
gain steps = round(0.017157 / 0.000977) = round(17.561)
gain steps = 18

The required offset steps are

$gain_{calib}$ [1] = 1.4902 + 0.000977 × (522 − 512) = 1.49997
$gain_{new}$ [1] = 1.4902 + 0.000977 × (522 + 18 − 512) = 1.517556
Δoffset = −0.013072 × 1.49997 / 1.517556 = −0.0129205
offset steps = round(−0.0129205 / 0.004) = round(−3.230)
offset steps = −3

6 Appendix

6.1 Python script for calibration calculations

```
   * Copyright 2015 NXP. NXP Confidential. This software is owned or
controlled by NXP and may only be
   * used strictly in accordance with the applicable license terms
found at
   * https://www.nxp.com/LA_OPT_NXP_SW. The "production use license"
in Section 2.3 in the NXP SOFTWARE
   * LICENSE AGREEMENT is expressly granted for this software.

   # -*- coding: utf-8 -*-

   def mean(values):
   """ calculates and returns the mean of the provided values
   :mean:  = 1 / n * sum(values)
   """
   mean = 0
   for x in values:
   mean += x
   mean = mean / len(values)
   return mean

   def slope(xvalues, yvalues):
   """ calculates and returns the slope m
   :slope: m = sum( (xn-xmean)*(yn-ymean)) / sum( (xn-xmean)**2 )
   """
   xmean = mean(xvalues)
   ymean = mean(yvalues)
   sum1 = 0
   sum2 = 0
   for n in range(len(xvalues)):
   sum1 += (xvalues[n]-xmean)*(yvalues[n]-ymean)
   sum2 += (xvalues[n]-xmean)**2
   m = sum1 / sum2
   return m

   def intercept(xvalues, yvalues):
   """ calculates and calculates the intercept
   b = ymean - m * x_mean
   """
   m = slope(xvalues, yvalues)
   xmean = mean(xvalues)
   ymean = mean(yvalues)
   b = ymean - m * xmean
   return b

   def linearEstimation(measured_values, reference_values):
   """ calculates the linearEstimation(measured, reference) and
returns slope and offset

   :returns: slope and offset
   """
   m = slope(reference_values, measured_values)
   b = intercept(reference_values, measured_values)
   return m, b
```

```
# ----------------------------------------------------------------------
# MM9Z1_638 Constants
# ----------------------------------------------------------------------

  shunt = 100e-6  #[V/A]

  def Shunt(voltage):
  """ returns the current for the voltage value provided
  """
  return voltage/shunt


  # this captures the information provided in Table 144. Acquisition channel compensation values of the data sheet
  #   channel name : (               offset details                   ,            gain details              ),
  #                   (   lsb           , def-value , def-code , unit )   (  lsb   , def-value , def-code , unit)
  mm9z1_638_acq_comp = {
  "VSENSE3"    : (( 3.998e-3         , 0.0       , 0x00     , 'V'  ),  (0.000488, 0.9922    , 0x200   , '1'  )),
  "VSENSE2"    : (( 2.000e-3         , 0.0       , 0x00     , 'V'  ),  (0.000488, 1.0679    , 0x200   , '1'  )),
  "VSENSE1"    : (( 1.000e-3         , 0.0       , 0x00     , 'V'  ),  (0.000488, 1.2207    , 0x200   , '1'  )),
  "VSENSE0"    : (( 1.000e-3         , 0.0       , 0x00     , 'V'  ),  (0.000488, 0.7632    , 0x200   , '1'  )),
  "VSENSEEXT"  : (( 0.100e-3         , 0.0       , 0x00     , 'V'  ),  (0.000488, 0.7632    , 0x200   , '1'  )),
  "ISENSE_G4"  : ((Shunt(25.602e-6)  , 0.0       , 0x00     , 'A'  ),  (0.000977, 1.4902    , 0x200   , '1'  )),
  "ISENSE_G16" : ((Shunt( 6.400e-6)  , 0.0       , 0x00     , 'A'  ),  (0.000977, 1.4902    , 0x200   , '1'  )),
  "ISENSE_G64" : ((Shunt( 1.600e-6)  , 0.0       , 0x00     , 'A'  ),  (0.000977, 1.4902    , 0x200   , '1'  )),
  "ISENSE_G256": ((Shunt( 0.400e-6)  , 0.0       , 0x00     , 'A'  ),  (0.000977, 1.4902    , 0x200   , '1'  )),
  "TSENSEEXT"  : (( 0.152e-3         , 0.0       , 0x00     , 'V'  ),  (0.000977, 1.0       , 0x80    , '1'  )),
  "TSENSEITS"  : (( 0.06384          , 0.0       , 0x00     , 'K'  ),  (0.000977, 1.0       , 0x80    , '1'  )),
  }


  def code2gain(code, channel):
  """ returns the gain value for a given code and channel

  :example:
  code2gain(512, 'VSENSE2')  returns  1.0679
  """
  _offsetinfo, gaininfo = mm9z1_638_acq_comp[channel]
  lsb, defvalue, defcode, _unit = gaininfo
  return defvalue + lsb * (code - defcode)


  def code2offset(code, channel):
  """ returns the offset value for a given code

  :example:
  code2offset(10, 'VSENSE2')  returns  0.020
  """
  offsetinfo, _gaininfo = mm9z1_638_acq_comp[channel]
  lsb, defvalue, defcode, _unit = offsetinfo
  return defvalue + lsb * (code - defcode)


  def lsbGain(channel):
  """ returns the lsb of the gain commpensation for the channel

  :example:
  lsbGain('VSENSE2')  returns  0.000488
  """
  _offsetinfo, gaininfo = mm9z1_638_acq_comp[channel]
  lsb, _defvalue, _defcode, _unit = gaininfo
  return lsb

  def lsbOffset(channel):
  """ returns the lsb of the gain commpensation for the channel

  :example:
  lsbOffset('VSENSE2')  returns  0.002
  """
  offsetinfo, _gaininfo = mm9z1_638_acq_comp[channel]
  lsb, _defvalue, _defcode, _unit = offsetinfo
  return lsb


  def calcSteps(channel, ifr, reference_values, measured_values, return_float=False):
  """ calculates the required compensation steps based on the inputs

  :return: gainsteps, offsetsteps    (return_float=False)

  for debugging:
  :return: gainstepsfloat, offsetstepsfloat    (return_float=True)

  """
  ifrvsg, ifrvo = ifr
  m, b = linearEstimation(measured_values, reference_values)

  gainerror = 1 - m
  offseterror = -b

  deltagain = gainerror * code2gain(ifrvsg, channel)
  gainsteps = round(deltagain / lsbGain(channel))
  gainstepsfloat = deltagain / lsbGain(channel)

  deltaoffset = offseterror * code2gain(ifrvsg, channel) / code2gain(ifrvsg+gainsteps, channel)
  offsetsteps = round(deltaoffset / lsbOffset(channel))
  offsetstepsfloat = deltaoffset / lsbOffset(channel)

  if return_float:
  return gainstepsfloat, offsetstepsfloat
  else:
  return gainsteps, offsetsteps


  if __name__ == '__main__':

  print()
  print("======================================================================")
  print("                      5.6 Example VSENSE2                             ")
  print("======================================================================")

  channel = "VSENSE2"
  ifrvsg = 518
  ifrvo  = -25
  reference_values = [4.9994, 7.9999, 11.999, 15.999, 19.999]
  measured_values =  [4.966, 7.9645, 11.9595, 15.956, 19.952]
```

```
    gainsteps, offsetsteps = calcSteps(channel, (ifrvsg, ifrvo), reference_values, measured_values)

    print("  channel:          ", channel)
    print("  ifrvsg:           ", ifrvsg)
    print("  ifrvo:            ", ifrvo)
    print("  reference_values: ", reference_values)
    print("  measured_values:  ", measured_values)
    print("  -> gainsteps:     ", gainsteps)
    print("  -> offsetsteps:   ", offsetsteps)

    # for debugging
    print("\nexact values for information only!")
    gainstepsfloat, offsetstepsfloat = calcSteps(channel, (ifrvsg, ifrvo), reference_values, measured_values,
return_float=True)

    print("  gain-error:       ", 1 - slope(reference_values, measured_values))
    print("  offset-error:     ", -1 * intercept(reference_values, measured_values))
    print("  gainsteps:        ", gainstepsfloat)
    print("  offsetsteps:      ", offsetstepsfloat)


    print()
    print("=========================================================================")
    print("                      5.7 Example ISENSE                                 ")
    print("=========================================================================")

    channel = "ISENSE_G256"
    reference_values = [ -38.000, -19.028, 10.052, 19.184, 38.140 ]
    measured_values = [-37.554, -18.798, 9.958, 18.975, 37.714]
    ifrvsg = 522
    ifrvo = 0       # no value for ISENSE
    gainsteps, offsetsteps = calcSteps(channel, (ifrvsg, ifrvo), reference_values, measured_values)


    print("  channel:          ", channel)
    print("  ifrvsg:           ", ifrvsg)
    print("  ifrvo:            ", ifrvo)
    print("  reference_values: ", reference_values)
    print("  measured_values:  ", measured_values)
    print("  -> gainsteps:     ", gainsteps)
    print("  -> offsetsteps:   ", offsetsteps)

    # for debugging
    print("\nexact values for information only!")
    gainstepsfloat, offsetstepsfloat = calcSteps(channel, (ifrvsg, ifrvo), reference_values, measured_values,
return_float=True)

    print("  gain-error:       ", 1 - slope(reference_values, measured_values))
    print("  offset-error:     ", -1 * intercept(reference_values, measured_values))
    print("  gainsteps:        ", gainstepsfloat)
    print("  offsetsteps:      ", offsetstepsfloat)
```

6.2 Output of Python script

```
=========================================================================
     5.6 Example VSENSE2

=========================================================================
     channel:          VSENSE2
     ifrvsg:           518
     ifrvo:            -25
     reference_values: [4.9994, 7.9999, 11.999, 15.999, 19.999]
     measured_values:  [4.966, 7.9645, 11.9595, 15.956, 19.952]
     -> gainsteps:     2
     -> offsetsteps:   14

     exact values for information only!
     gain-error:       0.0009181346061648554
     offset-error:     0.028459437224396922
     gainsteps:        2.014680827971926
     offsetsteps:      14.216760827598382


=========================================================================
     5.7 Example ISENSE

=========================================================================
     channel:          ISENSE_G256
     ifrvsg:           522
     ifrvo:            0
     reference_values: [-38.0, -19.028, 10.052, 19.184, 38.14]
     measured_values:  [-37.554, -18.798, 9.958, 18.975, 37.714]
     -> gainsteps:     18
     -> offsetsteps:   -3
```

```
exact values for information only!
gain-error:         0.011438086895173227
offset-error:       -0.013072264638248843
gainsteps:          17.560682907014318
offsetsteps:        -3.2301946006332094
```

7 Abbreviations

Table 18. Abbreviations

Acronym	Description
ACQ	acquisition of analog data using a sigma-delta analog-to-digital converter
ADC	analog-to-digital converter
AGC	automatic gain control
ATE	automated test equipment; used at the end of NXP manufacturing process
BMS	battery management system
Calibration	process performed to obtain compensation values
Compensation	method of compensating errors resulting in a more accurate behavior (measurement result)
CVMI	current and voltage measurement interrupt
Device	MM9Z1x638 integrated chip
Device calibration	device calibration handles calibration of the acquisition channels for device effects only (over temperature)
DMM	digital multimeter
EEPROM	electrically erasable programmable read-only memory
ETS	external temperature sensor
FLASH	read-only memory for program and data
GUI	graphical user interface
IBS	intelligent battery sensor
IFR	information row; special read-only area in the flash memory (ROM), used to store compensation values, etc.
IIR	infinite impulse response
IRC	internal RC
ISENSE	current measurement acquisition channel
ITS	internal temperature sensor
LIN-bus	local interconnect network bus; 12 V single wire automotive communication interface and standard
LSB	least significant bit; also used to indicated ADC resolutions, e.g. 1 mV/LSB
NTC	negative temperature coefficient; resistor, used as temperature sensing device
ODR	output data rate
PCB	printed-circuit board
PGA	programmable gain amplifier
Startup trimming	startup trimming is the process of copying ATE obtained trim values during the device startup

Acronym	Description
System calibration	calibration on system (e.g. device solder on PCB) level, including system effects like soldering, configuration and (partly) external components
TSENSE	temperature measurement acquisition channel
UPS	uninterruptible power supply
VSENSE	voltage measurement acquisition channel

8 References

[1] product summary page
http://www.nxp.com/MM9Z1_638

[2] battery management page
http://www.nxp.com/batterymanagement

9 Legal information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Suitability for use in automotive applications — This NXP Semiconductors product has been qualified for use in automotive applications. Unless otherwise agreed in writing, the product is not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — While NXP Semiconductors has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP Semiconductors accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

9.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

Tables

Figures

Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 5 June 2019
Document identifier: AN12301