EXCELITAS TECHNOLOGIES

CaliPile Application Note

I2C interface Description, Functional Parameter Setting and Event Determination

Version 1.2 - Date: 4.5.2016

Introduction

The EXCELITAS thermopile sensors are used for remote temperature sensing by the measurement of infrared (IR) radiation. In addition the CaliPile offers motion and presence event detection solely handled by the integrated signal processor, while the host microcontroller can be in power saving sleep mode. Upon detection of an event per its programmed configuration the CaliPile wakes up the host microcontroller by generating an interrupt.

Features and Benefits

- Low Supply Current
- Factory-calibration for remote temperature measurement
- Wake-up host microcontroller from Sleep Mode
- Programmable Detection Criteria
- Digital I2 interface
- SMD housing
- Low EMI susceptibility
- Four possible I2C addresses by A0 and A1 pin
- Open Drain interrupt output
- Integrated ambient temperature sensor

Applications

- Battery operated Devices
- Low Power Motion and Presence Detection
- Temperature Sensing
- Low Power Monitoring / Control Devices

1 Scope

The thermopile sensor with integrated signal processor is an event-detect-monitor with I2C interface and an interrupt output designed to interface with micro-controllers. The signal processor allows for very low system power consumption as it monitors the thermopile voltage and only generates a wakeup interrupt for the microcontroller when required.

This document gives detailed information about the I2C interface and how to configure the functional parameters to generate a wake up interrupt at the desired event.

2 I2C Interface Characteristics

An I2C serial interface is provided to read out the sensors temperature signals and for read and write access of EEPROM, configuration and status registers.

The following chapters give the complete information to understand and to operate the I2C interface of the CaliPile. For the complete I2C spec (version 2.1) refer to: **www.i2c-bus.org**

The SCL is a bidirectional input and output used as synchronization clock for serial communication. The SDA is a bidirectional data input and output for serial communication. The SCL and SDA outputs operate as open drain outputs only. External pull-up resistors are required. The pull-up resistor does all the work of driving the signal line high. All devices attached to the bus may only drive the SDA and SCL lines low.

The I2C interface allows connection of a master device (MD) and one or more slave devices (SD). The CaliPile can be operated as a SD only. The MD provides the clock signals and initiates the communication transfer by selecting a SD through a slave address (SA) and only the SD, which recognizes the SA should acknowledge (ACK), the rest of SDs should remain silent.

The general data transfer format is as follows:

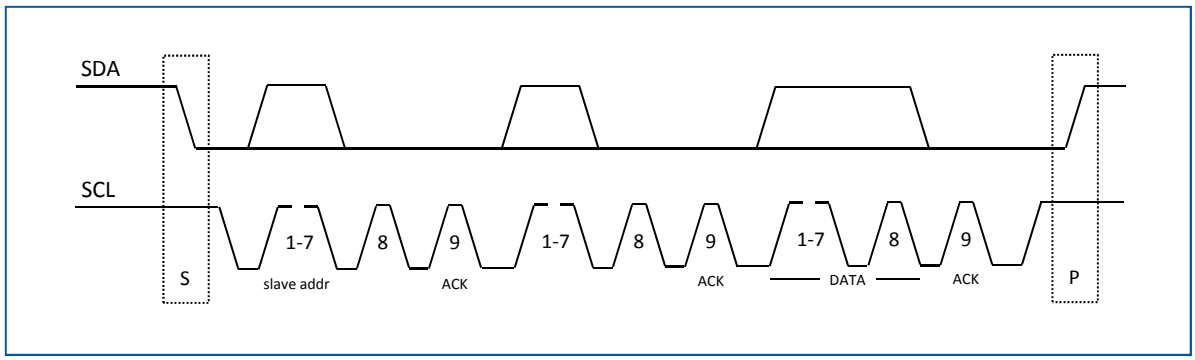


Figure 1: Data transfer format

	Start Condition
S	
R/W#	Read = 1 / Write# = 0
(N)ACK	(Not) Acknowledge; ACK = 0 , NACK = 1
P	Stop Condition

In idle state the SCL and SDA lines are both logical high. After the start condition S, the MD must place the 7-bit address of the SD it wants to address on the bus, followed by an eighth bit indicating the direction of the data transfer (R/W#). A data transfer is always terminated by a stop condition P generated by the MD.

The Most Significant Bit of every byte is transferred first.

After every 8 bits received by the SD an ACK/NACK takes place. The MD should stop the communication and repeat the message again if the SD NACKs a byte. If the MD has to receive data it has to generate ACK after transmission of each byte except for the last byte which must not be acknowledged by the MD to signal end of transmission.

The CaliPile does not support address resolution protocol (ARP) and bus arbitration. Therefore it is only suitable for single MD multiple SD bus topology and is not able to handle multiple SDs with similar SA.

The data on SDA must be stable during the high period of the clock. The data on SDA can only change when the SCL is low. The data is fetched by both MD and SDs on rising edge of the SCLK.

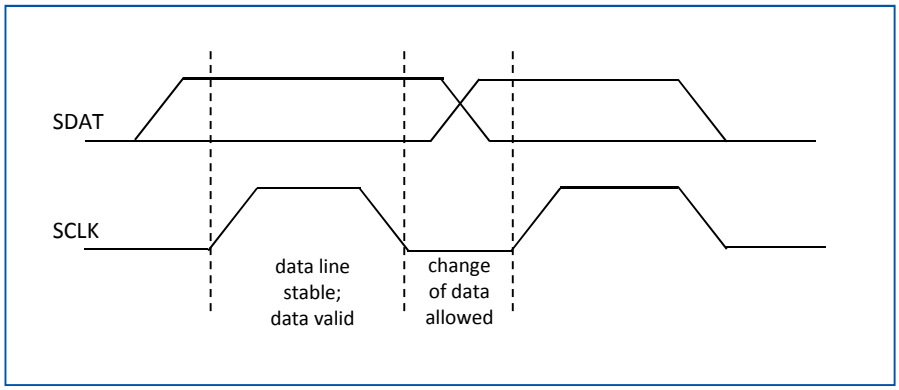


Figure 2: Data validity

2.1 START and STOP conditions

Two unique bus situations define a message START and STOP condition.

1. A high to low transition of the SDAT line while SCLK is high indicates a message START condition.

2. A low to high transition of the SDAT line while SCLK is high defines a message STOP condition. START and STOP conditions are always generated by the bus master. After a START condition the bus is considered to be busy. The bus becomes idle again after certain time following a STOP condition or after both the SCLK and SDAT lines remain high for more than $t_{HIGH:MAX}$.

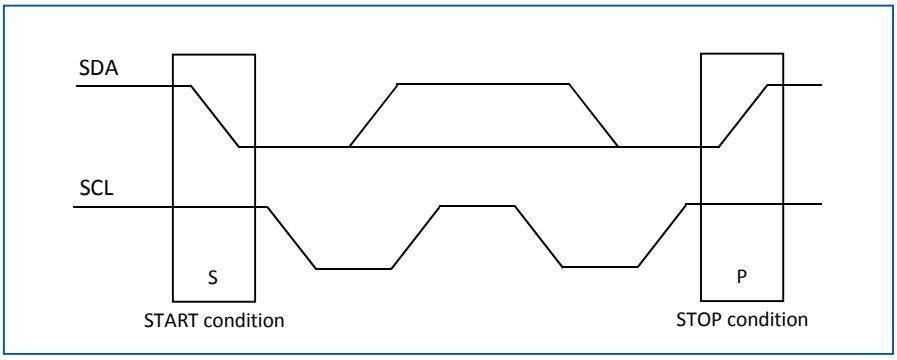


Figure 3: START and STOP Condition

2.2 Clock low extension

The CaliPile may need some time to process received data or may not be ready yet to send the next byte. In this case the SD can pull the SCL clock low to extend the low period of SCL and to signal to the master that it should wait. Once the clock is released the master can proceed with the next byte.

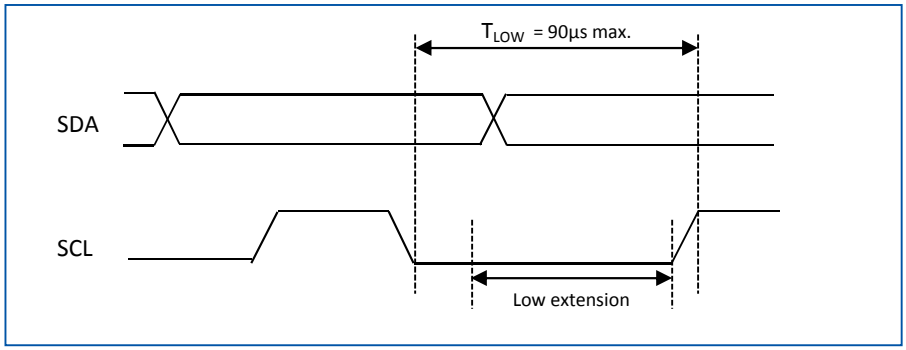


Figure 4: Clock low extension

2.3 Slave Address

After power up the CaliPile responds to the General Call Address (0x00) only. Upon receipt of a general call, it loads its slave address from EEPROM (ESA<7:0>). The slave address stored in the EEPROM consists of 7 address bits (6:0) and 1 address control bit (7). If the address control bit is set, the slave address read from the EEPROM is merged with the information from the slave address select pins A1 and A0.

Example:

ESA<7:0> = '1ABCDEFG', <A1:A0> ='YZ', resulting slave address = 'ABCDEYZ'.
ESA<7:0> = '0ABCDEFG', <A1:A0> ='YZ', resulting slave address = 'ABCDEFG'.

2.4 Data Format

Below is a key to the protocol diagrams of the following chapters.

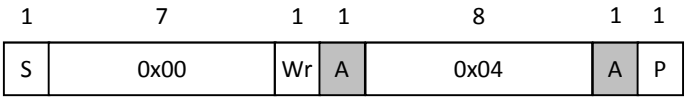
1	7	1	1	8	1	8	1	1
S	Slave Address	Rd	A	Register Address	A	Data Byte	A	P

S	Start Condition
Rd	Read (bit value of 1)
Wr	Write (bit value of 0)
A	ACK = Acknowledge (bit value of 0)
$\overline{A}$	NACK = Not Acknowledge (bit value of 1)
P	Stop Condition

	Master-to-Slave
	Slave-to-Master
…	Continuation of Protocoll

2.5 General Call

In order to re-fresh the slave address from EEPROM the MD has to send a general call (0x00) followed by the reload command (0x04). The slave may require up to 300µs for copying the slave address from EEPROM information into the register.

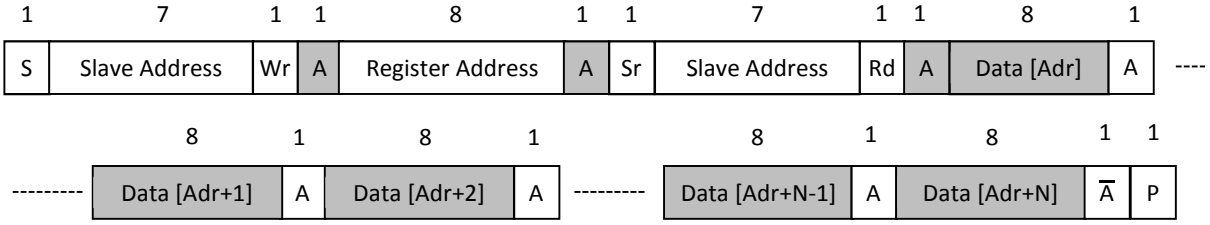


2.6 Reading Data from Register

Each register can be read through the I2C bus interface. The address information following Slave address points to the register to be read. The SD may require some time to load the data into the serial interface and therefore apply "clock stretching" after reception of the address byte. Once the data is ready for transmission to the MD, clockstretching will be released and the MD can clock out the data byte.

The address pointer on the SD will be automatically incremented to prepare for the next data byte to be fetched for transmission. The SD may apply "clock stretching" again to enforce a waiting time, before the next data byte is ready for transmission. The address pointer will wrap around to 0 once it exceeds address 63.

Reading of data can be interrupted by the MD at any time by generating a stop or a new start condition or a "not acknowledge".

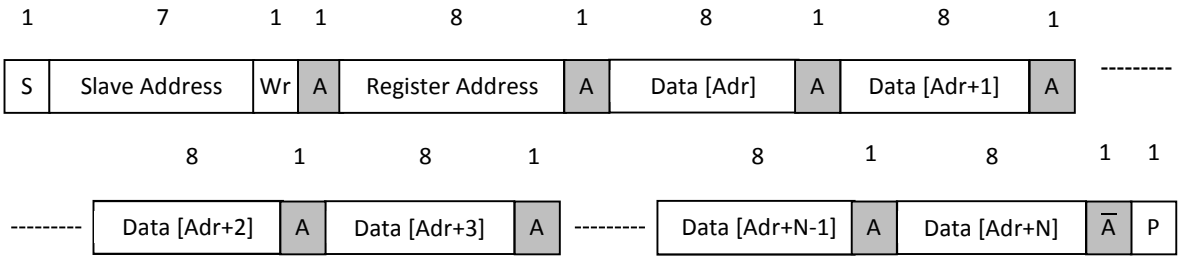


2.7 Writing Data to Register

Each register can be written to through the I2C bus interface. The address information following the Slave address specifies the location, where the next data byte is written to. The SD may require some time to write the data into the registers on chip and therefore apply "clock stretching" after reception of the data byte. Once the data is stored in the register, the slave will increment the address pointer and prepare for the next data byte to be received. The address pointer will wrap around when it exceeds 63.

Writing of data can be interrupted at any time by generating a stop or a new start condition or a "not acknowledge".

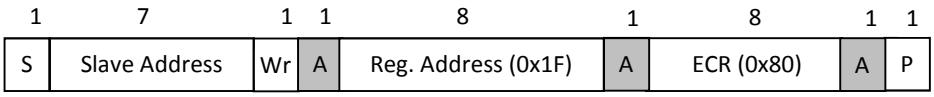
If the address points to a non-writable register, the register content remains unchanged.



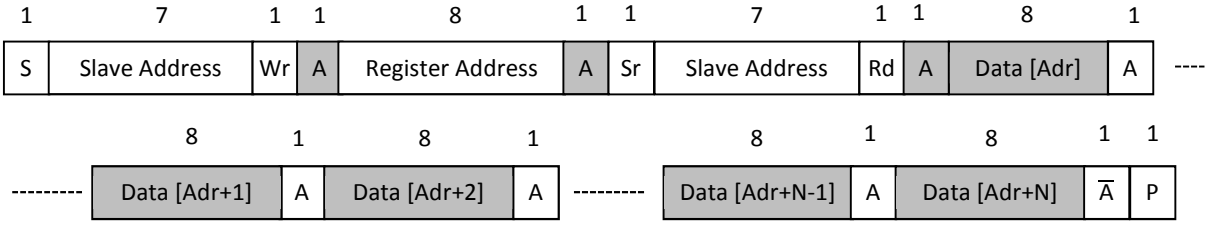
2.8 Reading EEPROM

A dedicated EEPROM control register (ECR) is provided to control access mode and to allow testing of EEPROM during production. Prior to reading EEPROM memory via I2C interface the control byte needs to be set accordingly. It is of importance to configure the EEPROM control register correctly as specified to ensure correct operation. In order to enable EEPROM reading, the ECR must be set to 0x80.

Note: Configuring the ECR for EEPROM read access causes increase of the supply current during EEPROM read operation until ECR will be set to 0x00 again.



Once the ECR has been setup correctly for read operation, the EEPROM cells can be addressed and read as follows.



The address information following the Slave address points to the EEPROM memory location to be read. The SD may require some time to load the data into the serial interface and therefore apply "clock stretching" after reception of the address byte. Once the data is ready for transmission to the MD, clock stretching will be released and the MD can clock out the data byte.

The address pointer on the SD will be automatically incremented to prepare for the next data byte to be fetched for transmission. The SD may apply "clock stretching" again to enforce a waiting time, before the next data byte is ready for transmission. The address pointer will wrap around to 0 once it exceeds address 63.

The EEPROM control register must be configured to 0x00 after the end of the EEPROM read operation to bring the supply current back to normal (lower) level.

3 Functional Parameter Setting and Event Determination

The CaliPile contains all functions required to allow an external microcontroller to detect activity and presence through a thermopile infrared detector. The parameters which should lead to a wakeup of the microcontroller can be programmed and adapted on the fly. The algorithm is based on various filter calculations of the sensor signals TP_{object} and $TP_{ambient}$, their differences and time derivatives. Aim is to minimize the power consumption of the system and allowing for high flexibility in adapting the algorithm to the actual conditions.

The basic function of the CaliPile is “presence detection”, “motion detection”, “ambient temperature shock detection” and “over temperature detection. Any of those functions can be selected by the host microcontroller as an interrupt source for wakeup. The parameters used to calculate the current state of “presence”, “motion” or “shock can be changed by the host controller through control registers. This allows the host controller to stay in sleep mode for most of the time and only be activated once the CaliPile detects a change which requires intervention.

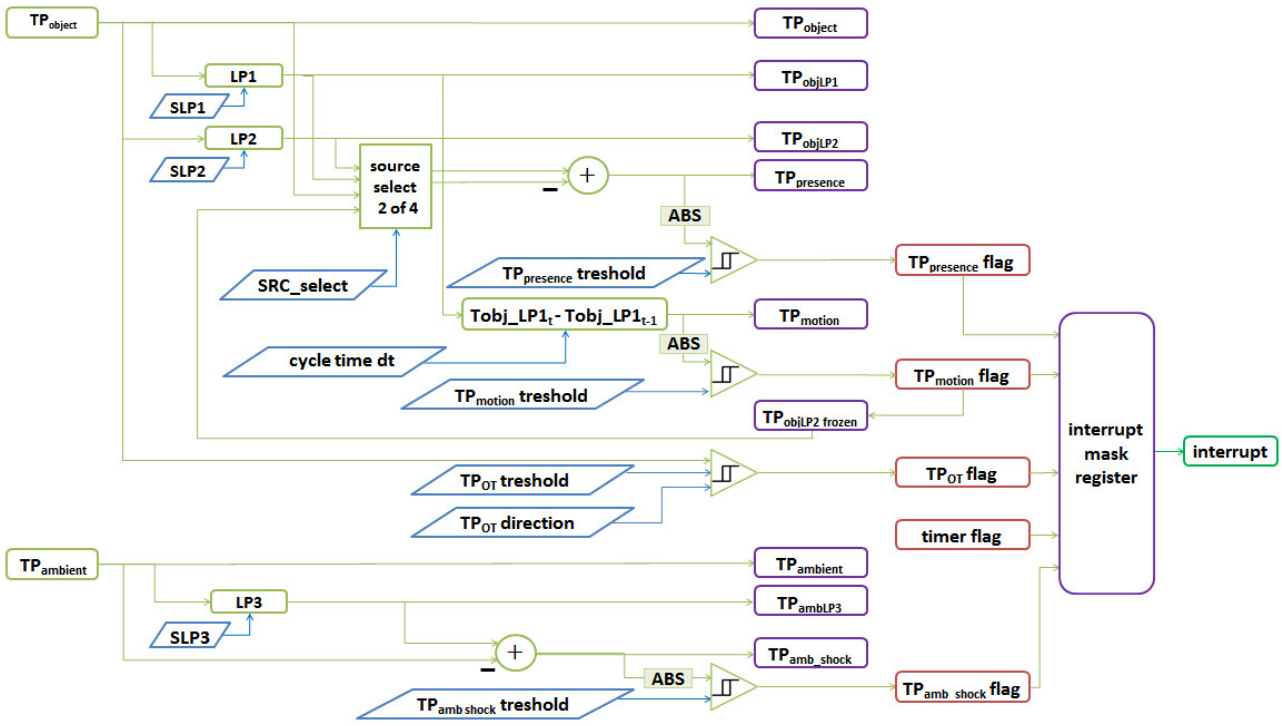


Figure 5: signal flowchart overview

3.1 Presence detection

Presence detection is accomplished by observing the difference between two user selectable signals which will be calculated from the thermopile raw signal T_{object}.
In order to achieve different speeds and noise levels for the results for presence detection, four signals are available for selection.

- The original signal TP_{object} from the thermopile output
- two signals, which have been processed by the low pass filters LP1 and LP2 with different user programmable time constants (S_{LP1}, S_{LP2}).

$$TP_{objLP1}(x) = TP_{object}(x) * S_{LP1} + TP_{objLP1}(x-1) * (1 - S_{LP1})$$
$$TP_{objLP2}(x) = TP_{object}(x) * S_{LP2} + TP_{objLP2}(x-1) * (1 - S_{LP2})$$

- The signal TP_{objLP2_frozen} which is the TP_{objLP2} output, that was saved at the moment the last motion event was detected

Thus various calculations for presence detection are possible and can be adapted to the actual conditions
e.g.:

$$TP_{presence} = TP_{object} - TP_{objLP2}$$
$$TP_{presence} = TP_{objLP1} - TP_{objLP2}$$
$$TP_{presence} = TP_{object} - TP_{objLP2\_frozen}$$
$$TP_{presence} = TP_{objLP1} - TP_{objLP2\_frozen}$$

The difference of the two selected signals will then compared with a programmable threshold $TP_{presence}treshold$.The $TP_{presence}flag$ is set once the difference of the two signals exceeds the threshold.

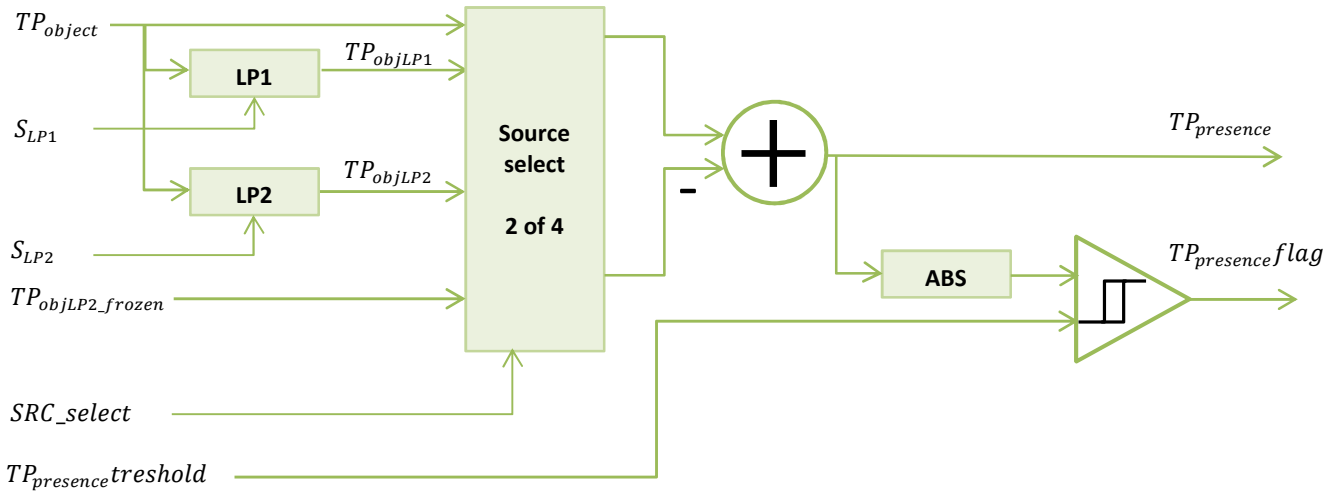


Figure 6: Presence detection algorithm flowchart

3.2 Motion detection

Motion detection is accomplished by observing the difference between two consecutive samples of $\mathrm{TP}_{\mathrm{objLP1}}$ with a programmable time interval dt. This is comparable to the 1st derivative of $\mathrm{TP}_{\mathrm{objLP1}}$.

$$TP_{motion} = \frac{dTP_{objLP1}}{dt}$$

The difference of the two signals will then compared with a programmable threshold $TP_{motion}threshold$. The $TP_{motion}flag$ is set once the difference exceeds the threshold.

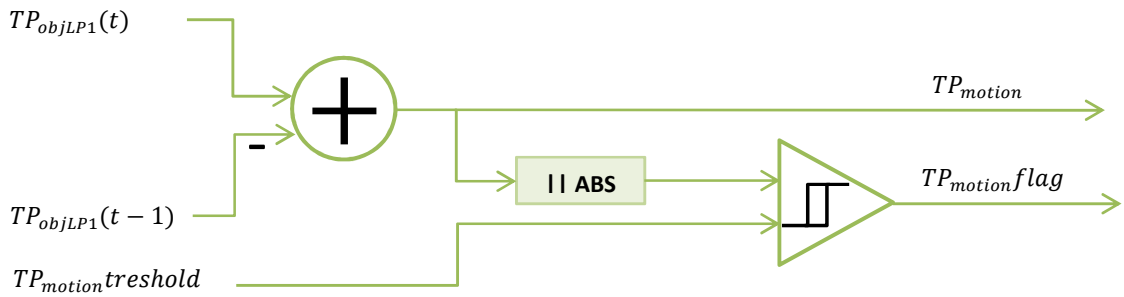


Figure 7: Motion detection algorithm flowchart

At the moment the $TP_{motion}flag$ is set, the current value of $\mathrm{TP}_{\mathrm{objLP2}}$ will be saved as $\mathrm{TP}_{\mathrm{objLP2\_frozen}}$ for further use in the presence detection algorithm.

3.3 Ambient temperature shock detection

Ambient temperature shock detection is accomplished by observing the difference between $\mathrm{TP}_{ambient}$ and the low pass filtered TP_{amb_LP3}.
The difference of the two signals will then compared with a programmable threshold $TP_{amb\_shock}treshold$. The $TP_{amb\_shock}flag$ is set once the difference exceeds the threshold.

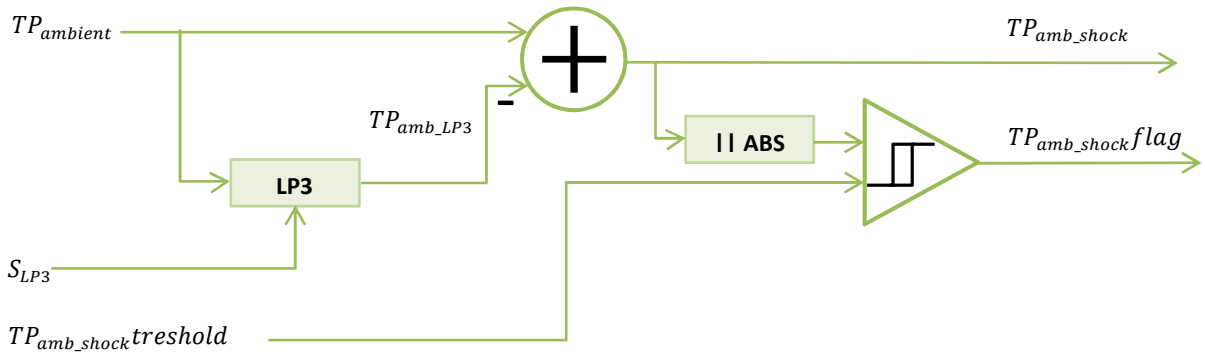


Figure 8: Ambient Temperature shock detection algorithm flowchart

3.4 Object temperature over or under limit detection

The raw data Tobj_Raw[16:1] is compared against the value specified in the object temperature threshold registers (Reg[28] MSB and Reg[29] LSB). An event is generated whenever the object temperature crosses the threshold. The user can select with the control register, which condition should lead to an interrupt – exceeding the limit or falling below the limit, Reg[26][4].
The interrupt is cleared when the µC reads the interrupt status register. A new interrupt can only be generated with a new event (object temperature crosses the limit).
To ensure correct system start up, the over temperature flag is set and the interrupt output is switched active after the device has been powered up. This feature is achieved with an on chip power on reset.
Reg[19][4] = actual status of the threshold comparator (0 = below threshold, 1 = above threshold).
Reg[18][4] = event - object temperatures crossed threshold (0 = from high to low, 1 = from low to high).
Reg[26][4] = selects, which event creates an interrupt (0 = from high to low, 1 = from low to high).

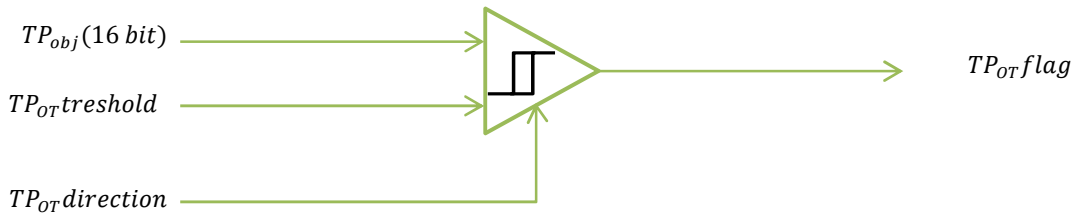


Figure 9: Object temperature over or under threshold limit algorithm flowchart

3.5 Hysteresis

The calculations for $TP_{presence}$, TP_{motion} and $TP_{amb\,shock}$ apply a hysteresis of 12.5%of the actual threshold value. The minimum hysteresis value is fixed to 5 counts. For the object temperature over/under limit detection TP_{OT} there is a fixed hysteresis of 64 counts built into the threshold comparator.
This is large enough to suppress the noise on the signals and to prevent false or frequent triggering of the corresponding flags if the signal is close to the threshold.

3.6 Control and Status Registers

The CaliPile contains control registers, which allows the external µC to configure certain parameters. Various registers allow the µC to gather data about history, events and actual sensor values.

Register #	Description	Size[bit]	Access
0	reserved	-	-
1-2,3[7]	TP_{object}	17	Read
3[6:0],4	$TP_{ambient}$	15	Read
5-7[7:4]	TP_{ObjLP1}	20	Read
7[3:0]-9	TP_{ObjLP2}	20	Read
10-11	TP_{ambLP3}	16	Read
12-14	TP_{ObjLP2_frozen}	24	Read
15	$TP_{presence}$	8	Read
16	TP_{motion}	8	Read
17	TP_{amb_shock}	8	Read
18[7:0]	Interrupt Status register	8	Read + autoclear
19[7:0]	Chip Status register	8	Read
20[3:0]	S_{LP1}	4	Write/Read
20[7:4]	S_{LP2}	4	Write/Read
21[3:0]	S_{LP3}	4	Write/Read
22	$TP_{presence}threshold$	8	Write/Read
23	$TP_{motion}threshold$	8	Write/Read
24	$TP_{amb\_shock}threshold$	8	Write/Read
25[4:0]	Interrupt Mask Register (over temperature, motion,presence,shock,timer)	5	Write/Read
26[1:0]	Cycle time for Motion detection differentiator	2	Write/Read
26[3:2]	SRC_select	2	Write/Read
26[4]	$TP_{OT}threshold\ direction$ 1 = Int active when Tobj goes above limit 0 = Int active when Tobj goes below limit	1	Write/Read
27[7:0]	Timer interrupt in 0.03sec	8	Write/Read
28,29	$TP_{OT}threshold$	16	Write/Read
30	reserved	-	-
31	EEPROM control byte	8	Write/Read
62:32	EEPROM,	248	Read
63	Slave address	8	Read

3.7 Control Register Details

Register# 1-2,3[7] : TPobject register

Register #1[7:0]								Register #2[7:0]								Register #3[7]
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7

Contains the 17 bit TPobject raw value in digits. This represents the current signal of the thermopile sensor element

Register# 3[6:0],4 : TPambient Register

Register #3[6:0]								Register #4[7:0]							
	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Contains the 15 bit TP ambient raw value in digits. This represents the current signal of the ambient temperature sensor .

Register# 5,6,7[7:4] : TP objLP1 Register

Register #5[7:0]								Register #6[7:0]								Register #7[7:4]			
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4

Contains the 20 bit TPobjLP1 value in digits. This represents the lowpass-filtered value of the Tobj_raw signal. The filter time constant for this filter stage can be set within register #20[3:0].

Register# 7[3:0],8,9 : TP objLP2 Register

Register #7[3:0]				Register #8[7:0]								Register #9[7:0]							
3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Contains the 20 bit TPobjLP2 value in digits. This represents the lowpass-filtered value of the TPobject signal. The filter time constant for this filter stage can be set within register #20[7:4].

Register# 10,11 : TPamb_LP Register

Register #10[7:0]								Register #11[0:7]							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Contains the 16 bit TPambLP value in digits. This represents the lowpass-filtered value of the TPambient signal. The filter time constant for this filter stage can be set within register #21[3:0].

Register# 12,13,14: TPobj_LP2_frozen Register

Register #12[7:0]								Register #13[0:7]								Register #14[4:7]							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Contains the 24 bit TPobjLP2_frozen value in digits. This copies the value of TPobjLP2 including four additional LSB digits acquired at the moment of the last valid motion event.

Register# 15 : TPobj_presence Register

Register #15[7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit TPobj_presence value in digits. TPobj_presence will be calculated as one of four different possible subtractions as given in the SCR_select register #26[3:2].See also chapter 9.1

Register# 16 : TPobj_motion Register

Register #16[7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit TPobj_motion value in digits. TPobj_motion will be calculated as difference between two subsequent values of Tobj_LP1. The time between these two points can be set within register #26[0:1].See also chapter 9.2

Register# 17 : TPamb_shock Register

Register #17[7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit TPamb_shock value in digits. TPamb_shock value will be calculated as difference between the raw TPambient signal and the filtered TPamb_LP signal.See also chapter 9.3

Register #18[7:0] : Interrupt Status Register

Bit[7:5] Interrupt status sign			Bit [4:0] Interrupt status flag				
7	6	5	4	3	2	1	0

The *Interrupt Status Register* copies the *Chip Status Register* in the moment an interrupt has been generated. This allows to read the conditions which lead to the interrupt even these conditions have been already changed. After the content of the Interrupt Status Register has been read it will be cleared automatically and the INT pin will be cleared (set to high).
Bit[4:0] contains the flags:[$TP_{OT}flag, TP_{presence}flag, TP_{motion}flag, TP_{amb\_shock}flag$, timer]
Bit[7:5] contains the sign: [$TP_{presence}sign, TP_{motion}sign, TP_{amb\_shock}sign$]

Register #19: Chip Status Register

Bit[7:5] Chip status sign			Bit [4:0] Chip status flag				
7	6	5	4	3	2	1	0

The Chip Status Register contains the flags of the five possible interrupt sources as well as the signs of the current values of $TP_{presence}$, $TP_{motion,}$ TP_{amb_shock} and the timer.
Bit[4:0] contains the flags: [$TP_{OT}flag, TP_{presence}flag, TP_{motion}flag, TP_{amb\_shock}flag$, timer]
Bit[7:5] contains the sign: [$TP_{presence}sign, TP_{motion}sign, TP_{amb\_shock}sign$]

The values for $TP_{presence}$ TP_{motion} and TP_{amb_shock} are always 8bit unsigned integers. Using their corresponding sign bits in register 19[7:5] it is possible to recalculate the data to signed values where [0] represents a positive value and [1] represents a negative value.

Example:

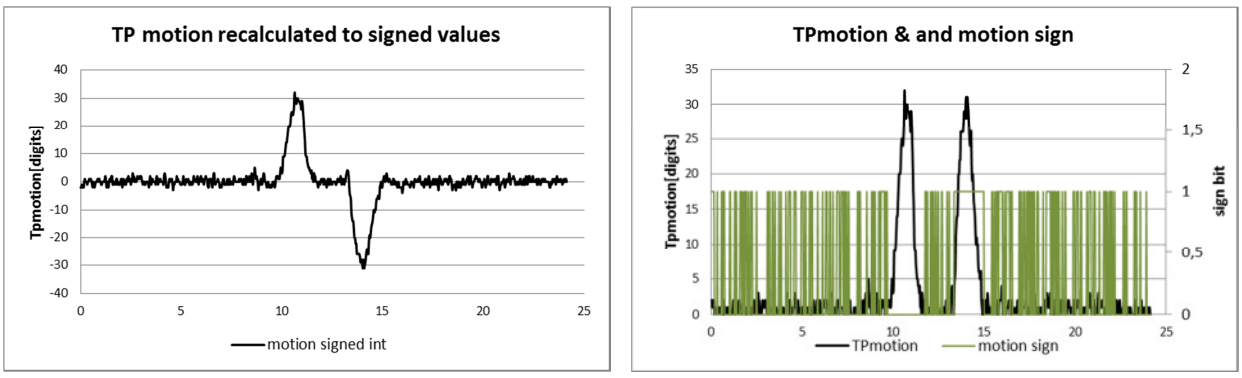


Figure 10: Original unsigned TPmotion data with corresponding sign bit and recalculated signed TPmotion data

Register #20: timeconstants for LP1 and LP2

Bit[7:4] S_{LP2}				**Bit [3:0] S_{LP1}**			
7	6	5	4	3	2	1	0

Register #21: timeconstant for LP3

Bit[7:4] not used				**Bit [3:0] S_{LP3}**			
				3	2	1	0

The registers #20 and #21[3:0] will be used to set the timeconstants for the three lowpass filters LP1, LP2 and LP3. For each of the three lowpassfilters twelve timeconstants/ cut-off frequencies can be selected.

Cut off [Hz]	**Select code [HEX]**	**Select code [BIN]**	**Cut off [Hz]**	**Select code [HEX]**	**Select code [BIN]**
$6{,}4*10^{-1}$	D	1101	$9{,}9*10^{-3}$	5	0101
$3{,}2*10^{-1}$	C	1100	$4{,}9*10^{-3}$	4	0100
$1{,}5*10^{-1}$	B	1011	$2{,}5*10^{-3}$	3	0011
$7{,}9*10^{-2}$	A	1010	$1{,}2*10^{-3}$	2	0010
$3{,}9*10^{-2}$	9	1001	$6{,}2*10^{-4}$	1	0001
$1{,}9*10^{-2}$	8	1000	$3{,}1*10^{-4}$	0	0000

Register# 22 : TPpresence threshold Register

Bit[7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit threshold value for $TP_{presence}$ in digits. Once the $TP_{presence}$ signal passes this threshold the corresponding presence flag will be set in the chip status register #19[3]

Register# 23 : TPmotion threshold Register

Bit [7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit threshold value for TP_{motion} in digits. Once the TP_{motion} signal passes this threshold the corresponding presence flag will be set in the chip status register #19[2]

Register# 24 : TPamb_shock threshold Register

Bit [7:0]							
7	6	5	4	3	2	1	0

Contains the 8 bit threshold value for TP_{amb_shock} in digits. Once the TP_{amb_shock} signal passes this threshold the corresponding presence flag will be set in the chip status register #19[1]

Register# 25[4:0] : Interrupt Mask Register

Bit[7:5] not used			Bit [4:0]				
			4	3	2	1	0

Contains the 5 bit mask value to activate the external interrupt output INT based on five different possible sources in the chip status register #19[4:0].

The INT pin will be activated only if the corresponding mask flag inside the interrupt mask register is set and the corresponding interrupt occurs as signaled in the chip status register 19 Bit[4:0].
Bit[4]:set to 1 activates the INT pin if the $TP_{OT}flag$ in register 19 has been set
Bit[3]:set to 1 activates the INT pin if the $TP_{presence}flag$ in register 19 has been set
Bit[2]:set to 1 activates the INT pin if the $TP_{motion}flag$ in register 19 has been set
Bit[1]:set to 1 activates the INT pin if the $TP_{amb\_shock}flag$ in register 19 has been set
Bit[0]:set to 1 activates the INT pin if the timer flag in register 19 has been set
If more than one mask bit has been set the INT pin will be activated for whatever flag in the chip status register comes first (OR condition).
The INT output will remain active until the µC reads the interrupt status register[18]. Interrupts are set when conditions change from inactive to active.

Register# 26[4:0] cycle time , SRC selectRegister & over temperature direction

Bit[7:5] not used			Bit[4] OT	Bit [3:2] SRC select		Bit [0:1] cycle time	
			4	3	2	1	0

As described in chapter 9.2 the TPmotion signal will be calculated as difference of two subsequent TPobj_LP1 values. The time between these two points can be set with a two bit value for Bit[0:1].
00 = 30ms, 01 = 60ms, 10 = 120ms, 11 = 240ms
Bit[2:3] allows to switch the signal sources to be used for TP presence calculation as explained in chapter 9.1.
00 = $TP_{presence} = TP_{object} - TP_{objLP2}$
01 = $TP_{presence} = TP_{objLP1} - TP_{objLP2}$
10 = $TP_{presence} = TP_{object} - TP_{objLP2\_frozen}$
11 = $TP_{presence} = TP_{objLP1} - TP_{objLP2\_frozen}$
Bit[4] allows to select which event creates an interrupt (1= interrupt if signal exceeds the OT threshold, 0=interrupt if signal falls below OT threshold)

Register# 27 : timer interrupt Register

Bit [7:0]							
7	6	5	4	3	2	1	0

Allows a timer overrun interrupt from 30ms up to 7.7s in 30ms steps

Register# 28,29 : TP_{OT} threshold Register

Register #28[7:0]								Register #29[7:0]							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Contains the 16 bit object temperature threshold value for over/under limit detection in digits.

Register# 31 : EEPROM control Register

Bit [7:0]							
7	6	5	4	3	2	1	0

Used to control EEPROM read actions. Refer to chapter 2.8.

4 Liability policy

Excelitas shall not be liable for direct, indirect, incidental, or consequential damages, including, but not limited to, loss of use, revenue, or prospective profits resulting from the use of this document or the product to which it relates. All warranties express or implied, including, but not limited to, warranties of merchant ability and fitness for a particular purpose are hereby disclaimed.

5 Copyright

6 More information

For additional information on Excelitas products, please visit our website at http:// www.excelitas.com.

Excelitas Technologies
22001 Dumberry Road
Vaudreuil-Dorion,
Quebec
Canada J7V 8P7
Telephone: (+1)
450.424.3300
Toll-free: (+1)
800.775.6786
Fax: (+1) 450.424.3345
detection@excelitas.com

European Headquarters
Excelitas Technologies
GmbH & Co. KG
Wenzel-Jaksch-Str. 31
D-65199 Wiesbaden
Germany
Telephone: (+49) 611 492 430
Fax: (+49) 611 492 165
detection.europe@excelitas.com

Excelitas
Technologies
Singapore, Pte.Ltd.
8 Tractor Road
Singapore 627969
Telephone: (+65)
6775 2022
Fax: (+65) 6778 1752