TLE4801

User manual



This user manual provides a comprehensive guide to the digital programming interface and configuration process of the TLE4801 sensor IC. The configuration parameters are stored in a customer multi-time programmable (CMTP) register, which is detailed in the register map section of this document.

Scope and Limitations

The electrical and timing parameters specified in this document are applicable directly to the sensor pins. External factors such as circuitry, programming equipment, and environmental influences are not considered.

Additional Resources

For a detailed understanding of the TLE4801's general behavior and top-level block diagram, please refer to the TLE4801 data sheet.

This document is intended for anyone who needs to program and calibrate the programmable IC for inductive sensing with SENT/SPC interface.

The table below lists the devices covered by this document.

Table 1 Related devices

Name	Description and Features	Package	Marking
TLE4801S16-S0000	Torque, Angle sensing: SENT	PG-TSSOP-16	4801S
TLE4801C16-S0000	Torque, Angle sensing: SPC	PG-TSSOP-16	4801C

Table of contents

	Table of contents	2
1	**Application circuits**	4
2	**Programming**	6
2.1	Programmer connection	6
2.2	Interface	7
2.2.1	Bit Encoding	7
2.2.1.1	Bit encoding programmer	7
2.2.1.2	Bit encoding sensor	7
2.2.1.3	Communication	8
2.2.2	Protocol	9
2.2.2.1	Frame description	9
2.2.2.2	Read access	9
2.2.2.3	Write access	10
2.2.3	Command and response frame	11
2.2.3.1	Command frame and CMD options	11
2.2.3.2	Unlock command	11
2.2.3.3	Write commands	12
2.2.3.4	Read command	13
2.2.3.5	OFF command	13
2.2.3.6	Response frame	13
2.2.4	SICI Interface electrical and programming timing characteristics	14
2.3	Programming flow and procedure	16
2.3.1	Programming flow	16
2.3.2	Programming procedure	17
2.4	Register map	19
2.4.1	Definition	19
2.4.2	CMTP Register map	20
2.4.3	CMTP Register description	21
2.4.4	Working registers	24
2.4.5	CMTP Status Register	26
2.4.6	Hardware status register	26
2.4.7	ADC status registers overview	29
2.4.8	Temperature register ADC	30
2.5	Built in self test - BIST	32
2.6	CRC	33
3	**Calibration**	36
3.1	Angles calibration	36
3.1.1	Parameters for calibration	36
3.1.2	Calibration flow	36

Table of contents

3.1.2.1 Offset compensation . . . 37
3.1.2.1.1 Offset calculation . . . 38
3.1.2.2 Gain compensation . . . 38
3.1.2.2.1 Gain calculation . . . 38
3.1.2.3 Non orthogonality compensation . . . 39
3.1.2.3.1 Non orthogonality compensation calculation . . . 39
3.1.2.4 Vector length . . . 40
3.1.2.5 Inverting rotation . . . 40
3.1.3 Torque calibration . . . 40
3.1.3.1 Difference angle uniqueness . . . 42
3.2 Lookup table . . . 44
3.2.1 1D lookup table . . . 44
3.2.1.1 1D lookup tables for dual linear angle output . . . 44
3.2.1.2 Mechanical angle error compensation with 1D look up tables . . . 46

4 Timing model . . . 50
4.1 Timing SENT/SPC . . . 50
4.2 SICI delay timing model for calibration procedures . . . 51

5 Parameter mapping . . . 52
5.1 Short serial message items . . . 52
5.1.1 Message IDs . . . 52
5.1.2 Temperature mapping . . . 52
5.2 Angle/Torque read out mapping . . . 52

6 Abbreviations . . . 54

7 References . . . 55

8 Revision History . . . 56

Disclaimer . . . 57

1 Application circuits

The application circuits in this chapter show the various communication possibilities of the TLE480x.

Interfaces overview:

- SENT: Angle information and diagnostic interface.
- SICI: Programming interface.

External components:

- A decoupling capacitor C_S of 100nF must be located as close as possible to the sensor supply pin.
- A buffer capacitor C_{BUF} of typ. 100nF must be located as close as possible to the VCBUF pin.
- C_L represents the maximum capacitance of the line (e.g. 50pF/meter)
- A pull-up resistor R_P. The maximum value is determined by the unit time and the total capacitance of the line. The minimum value is determined by the V_{OL} specification.
- Values in the application circuit are typical values from SAE J2716 SENT Filter.
- C_{L1} and C_{L2} capacitors depend on the T_X coil excitation frequency.
- Depending on application EMC requirements we recommend filter capacitors ($C0_A$ to $C3_B$) on each Rx-pin to improve EMC behaviour.

Coils:

- For applications using two sets of receiving coils (2x SIN, 2x COS). The application diagram below applies.
- For applications using only one set of receiving coils (1x SIN, 1x COS). The remaining pins have to be shorted.

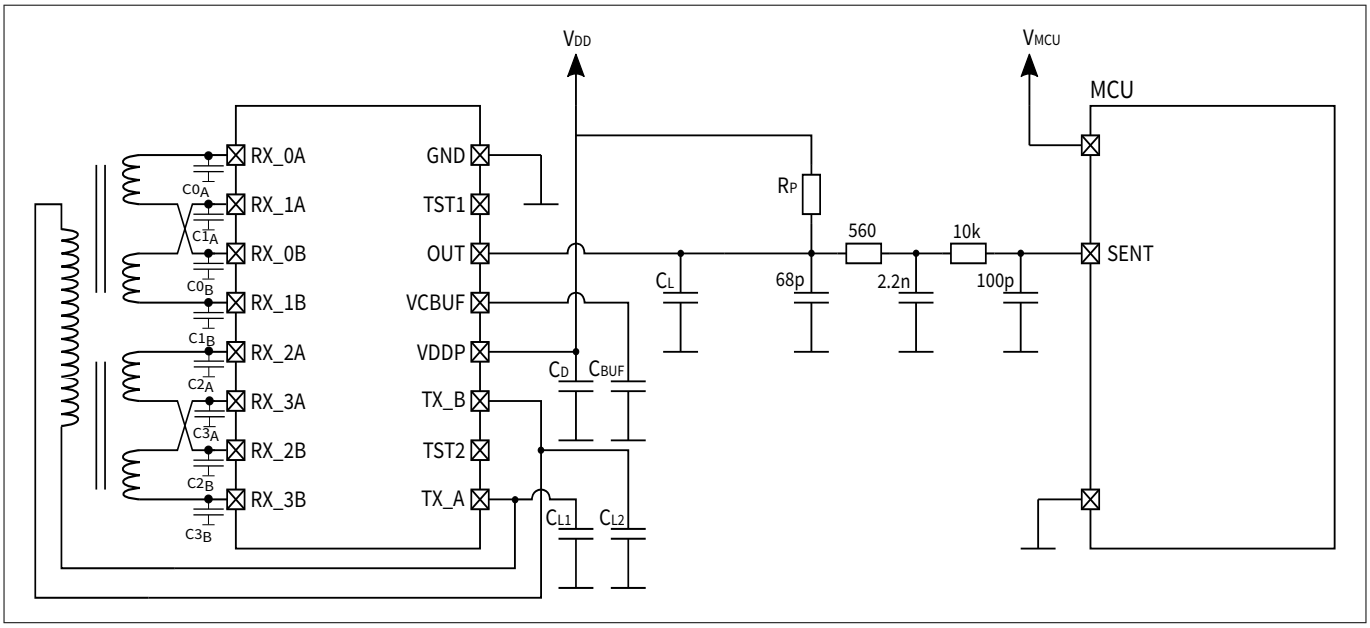

Figure 1 Application circuit for SENT interface

Interfaces overview:

- SPC: Angle information and diagnostic interface.
- SICI: Programming interface.

External components:

- A decoupling capacitor C_S of 100nF must be located as close as possible to the sensor supply pin.
- A buffer capacitor C_{BUF} of typ. 100nF must be located as close as possible to the VCBUF pin.
- C_L represents the maximum capacitance of the line (e.g. 50pF/meter)
- A pull-up resistor R_P. The maximum value is determined by the unit time and the total capacitance of the line. The minimum value is determined by the V_{OL} specification.

1 Application circuits

- C_{L1} and C_{L2} capacitors depend on the T_X coil excitation frequency.
- Depending on application EMC requirements we recommend filter capacitors ($C0_A$ to $C3_B$) on each Rx-pin to improve EMC behaviour.

Coils:

- For applications using two sets of receiving coils (2x SIN, 2x COS). The application diagram below applies.
- For applications using only one set of receiving coils (1x SIN, 1x COS). The remaining pins have to be shorted.

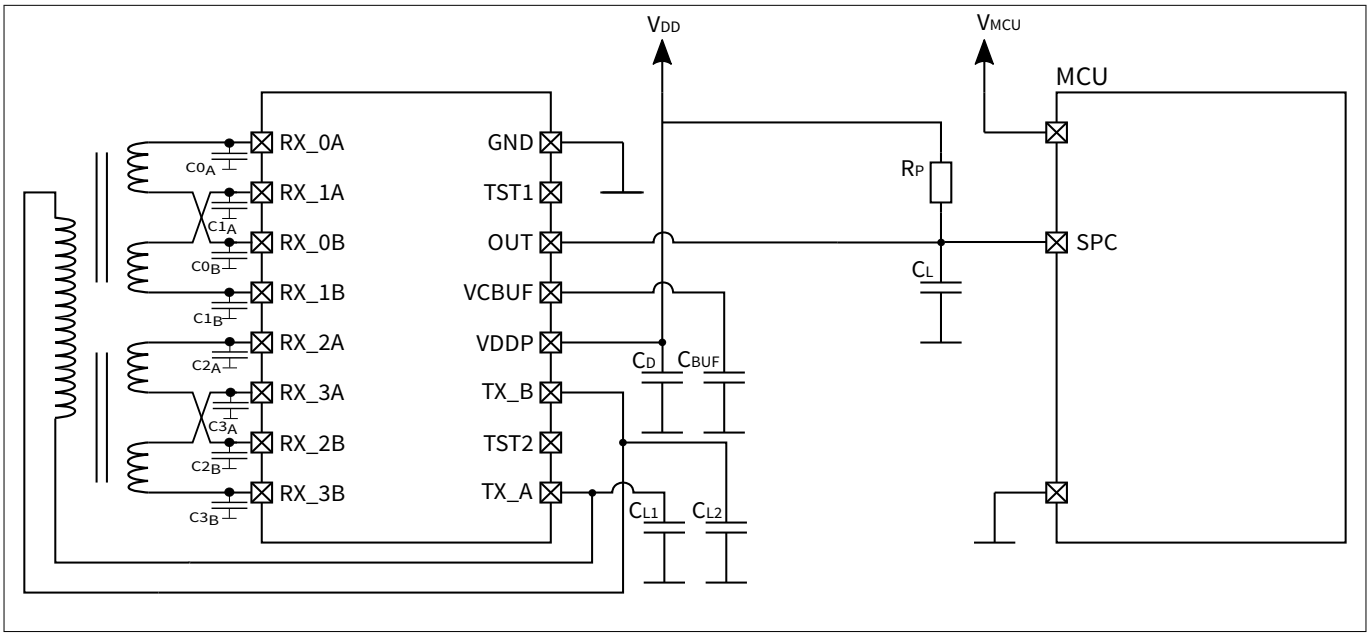

Figure 2 Application circuit for SPC interface

2 Programming

2.1 Programmer connection

The application circuits below show the typical connection between the TLE480x and the programmer. Two use-cases are described. In Application circuit for SICI (Figure 3) a standard microcontroller (I/O: 5V) is used to program the TLE480x. In Application circuit for SICI (Figure 4) a dedicated programmer able to handle a high voltage (I/O: V_{PROG}) on the SICI line is required. In either case, to perform a write, V_{PROG} must be applied on the TLE480x V_{DD} pin. (All parameters mentioned here can be found in the referenced datasheet: V_{DD},V_{PRO})

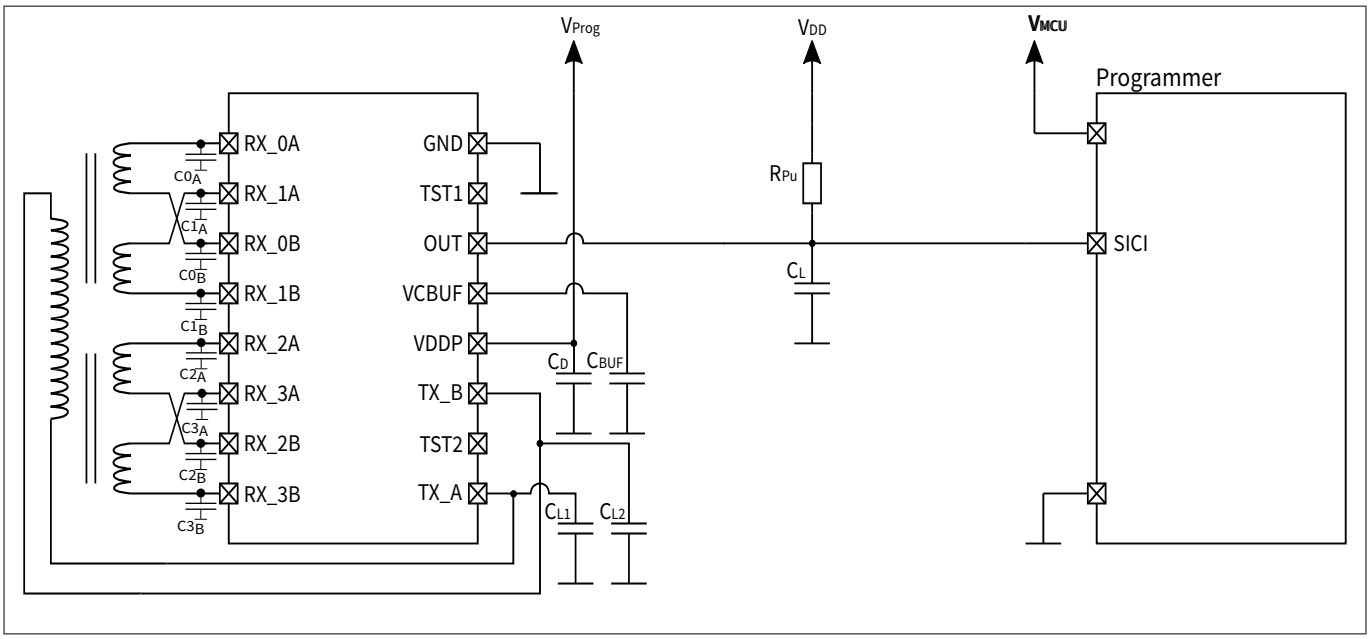

Figure 3 Application circuit for SICI (SICI line on V_{DD})

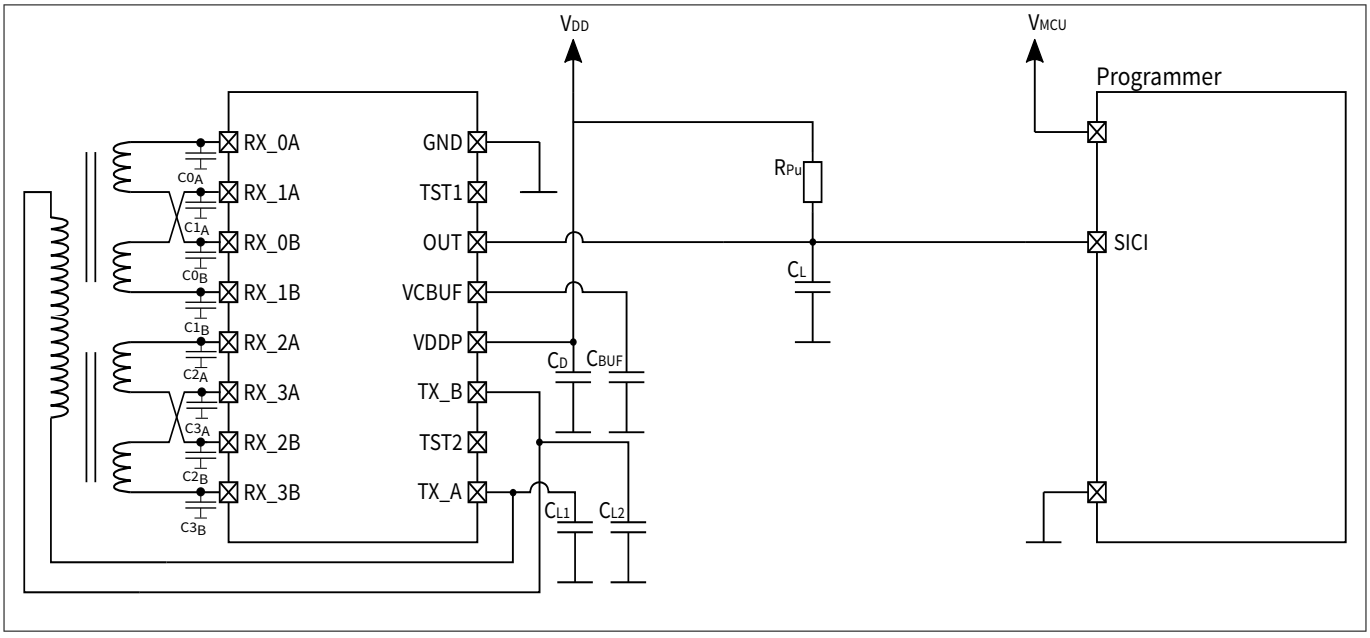

Figure 4 Application circuit for SICI (SICI line on V_{PROG})

2.2 Interface

The IC allows end-of-line programming of the CMTP and read access to all internal registers via the SICI (Serial Inspection and Configuration Interface) interface. The bidirectional communication mode can be entered during a limited time window directly after power on by sending the “activate programming mode” command. The SICI communication is based on transmitting a single bit to the sensor and immediately receiving a bit. These bits form a 18-bit word. It makes the interface bit-synchronous, robust and very flexible in timing. Bidirectional data transmission can thus be realized using only a single wire without a trimmed oscillator on sensor or programmer side. To enhance the robustness of the interface, the sensor repeats the 16-bit command word, sent by the programmer, in order to have a confirmation of the correct understanding of the message. Additionally, a time-out feature is implemented: If during a communication the sensor does not receive a bit from the programmer within a certain time span (please refer to table 27 in the datasheet under chapter References), the interface is reset and a new command can be sent.

2.2.1 Bit Encoding

2.2.1.1 Bit encoding programmer

Bits from the programmer are encoded as the difference in duration of consecutive "high" and "low" voltage levels or in other words as the duty-cycle of the PWM-modulated bits. To transmit a bit to the IC, the programmer sends one single high/low PWM signal with a period *T*. The logic value of the bit is then encoded as the difference between “high” and “low” time:

- to transmit a “0” to the IC, the programmer drives the line “low” for a short time t_{1_0}, then releases it to “high” for a long time t_{2_0} (typically t_{1_0}=0.33*T).
- to transmit a “1” to the IC, the programmer drives the line “low” for a long time t_{1_1}, then releases it to “high” for a short time t_{2_1}(typically t_{1_1}=0.67*T).

The IC recognizes the total bit-time interval *T* as the duration between two consecutive falling edges from the programmer for standard voltage I/O mode, and rising edges for standard inverted voltage/current mode. The bit encoding scheme is illustrated in below figure.

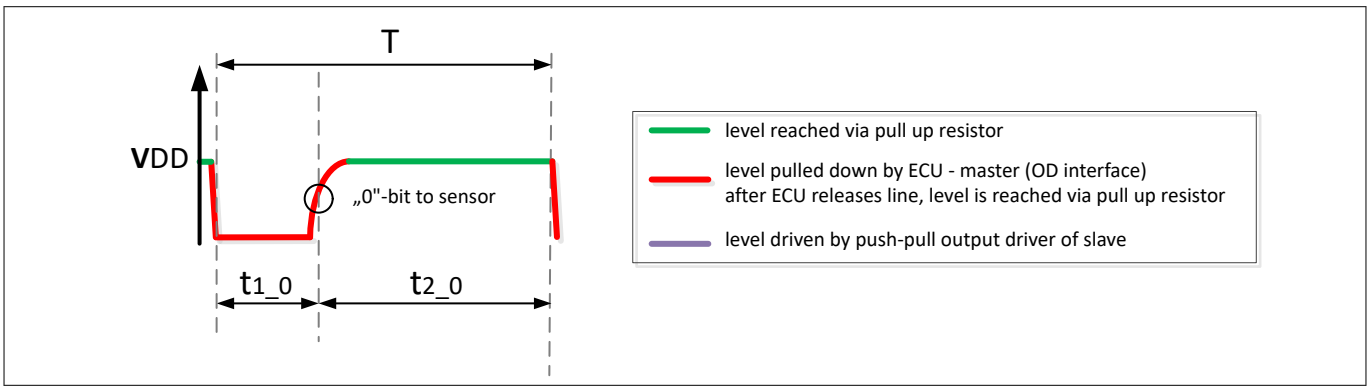

Figure 5 Timing description of programmer bit

2.2.1.2 Bit encoding sensor

After receiving a bit from the programmer, the IC answers by transmitting one bit:

- the IC transmits a “1” by pulling the SICI-line “low” for the aforementioned time.
- the IC transmits a “0” by releasing the line to "high" within the aforementioned time.

To read the bit transmitted by the TLE480x, the programmer has to check the level of the line after t_r (programmer low pulse) but before time t_4 (difference between programmer low and high level) has expired.

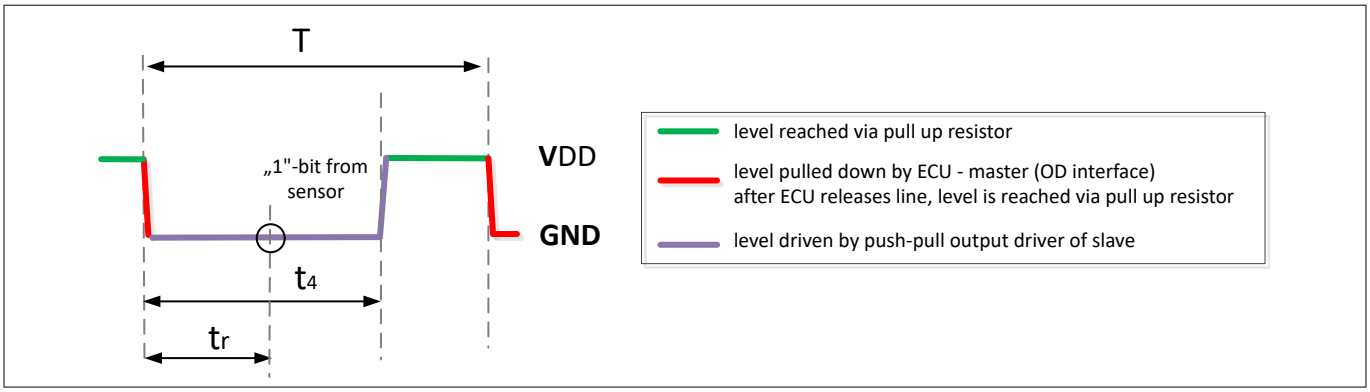

Figure 6 Timing description of response bit from sensor.

2.2.1.3 Communication

After receiving the response bit from the TLE480x, the programmer can pause for a time before transmitting the next bit. This timing has to be shorter than the time-out limit.

Otherwise, the interface is reset.

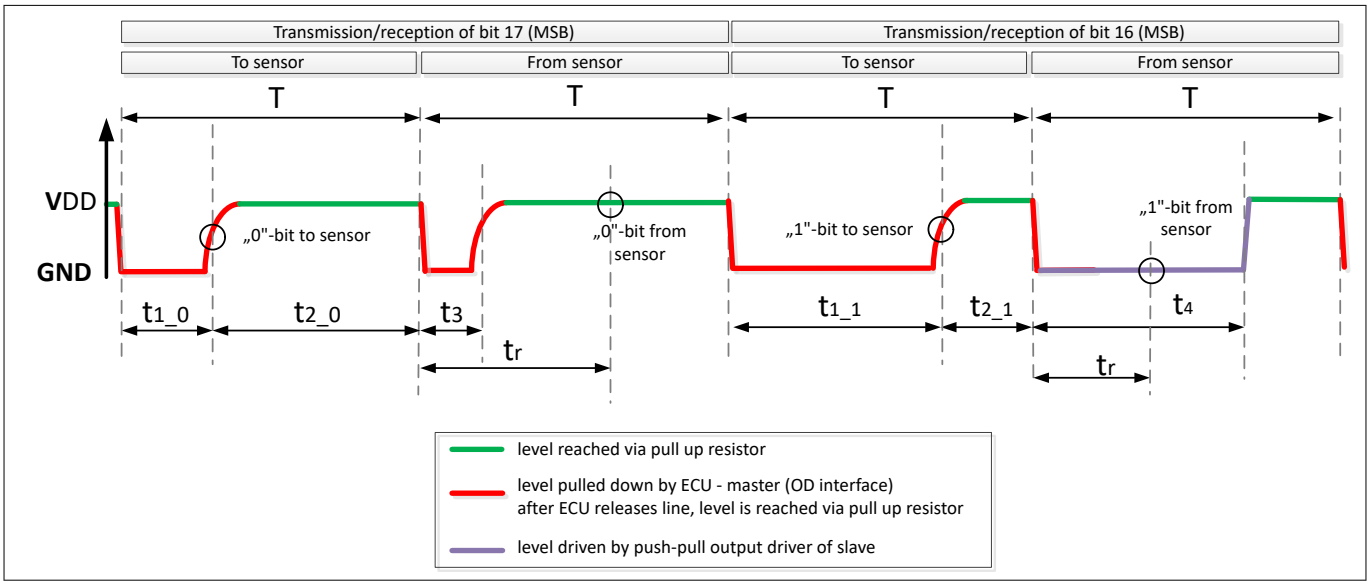

Figure 7 SICI communication example

The transmission rate of the interface is determined by the width T of the PWM signal sent by the programmer. A maximum suggested transmission speed of 25 kbit/s can be reached for T = 40 µs. Lowering the transmission rate might be required with a high capacitive load on the line.

Theoretically, with an optimal low capacitive load (no cable setup), a speed up to 250 kbit/s is possible. In this case the PWM signal period equals T = 2 µs.

2.2.2 Protocol

2.2.2.1 Frame description

A SICI frame consists of 1 control bit, 16 data bits arranged (MSB first), and 1 parity bit (odd parity). The SICI communication is started by the programmer sending an “activate programming mode” (has to be performed within $t_{activate}$) step to command directly after the power-up of the TLE480x. The TLE480x remains silent until the correct password is received. The control bit is “1” for command frames and “0” for data frames. The TLE480x is ready for programming 300 µs after receiving the “activate programming mode” command. After entering programming mode, the programmer can start transmitting one or multiple frames. Between two frames, the programmer has to pause for at least t_{pause}.

SICI is bit-synchronous, so every bit sent by the programmer is immediately responded by one bit from the TLE480x.

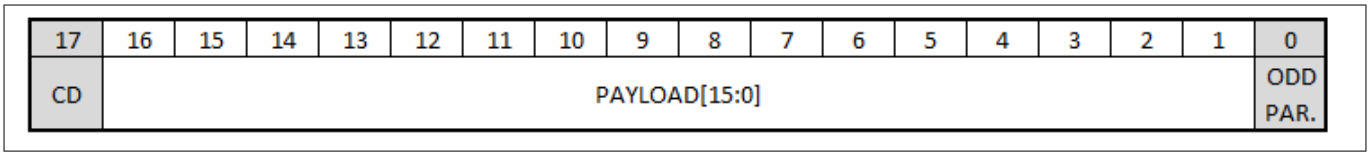

Figure 8

Bits	Name	Description
17	CD	Indicates if it is a command (if set to 1) or a data frame (if set to 0)
16:1	PAYLOAD	See Data/Command frame description
0	PAR	Odd parity over the whole frame

Remark:

Programming mode means when **siciunlock (0xE000)** command, as described in the programming example in *chapter 1.3.1*, is sent.

2.2.2.2 Read access

To read data content from the registers, the programmer must send a “read” command including the register address to read from. The "read" command from the programmer could be followed by one or multiple "dummy" data frames (the payload doesn't matter as long as the CD bit indicates the frame as a data frame) if multiple consecutive registers should be read out (burst access with automatic address increment). Otherwise, the programmer can also immediately send another read command for a different address while receiving the response for the previous read command in the same frame. In case the previous command did not request a response from the device, the SICI will respond with the previously received frame content for the payload bits.

The figure below shows the sequence of frames for reading multiple consecutive registers by sending one read command followed by multiple dummy data frames. The IC automatically increments the register address after every received data frame and responds by sending the content of the consecutive registers. A new command frame after the sequence of dummy data frames will then also contain the response for the last register readout.

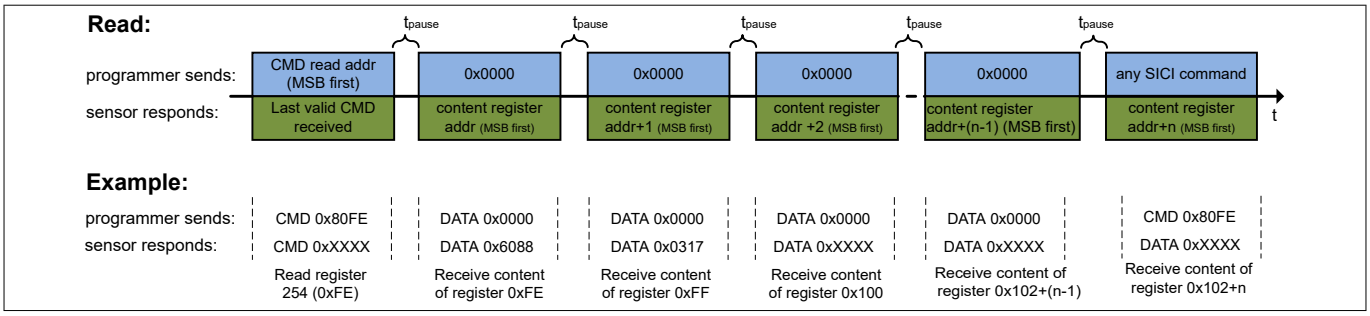

Figure 9 Frame order for read commands

2.2.2.3 Write access

To write data to registers, the programmer can send a “write” command including the register address to write to, followed by a data frame containing the data which shall be written to the addressed register. The TLE480x responds to the write command by showing the last valid command that it received, in case the last valid command was “activate programming mode” command the TLE480x will then reply with a “1” at the parity bit.

Otherwise, the TLE480x responds by repeating every bit of the last valid command it received, and it responds to the data frame by repeating the write command again.

The following procedure is necessary to perform a valid programming flow and to detect any programming issues. The read out of the OTP status registers(***254, 255*** ,*incremental read-out*) after writing each line is necessary and not optional. The right content of the OTP status register after a line has been programmed is as follows: **0x6088** *(register 254) and* **0x0317** *(register 255)*.

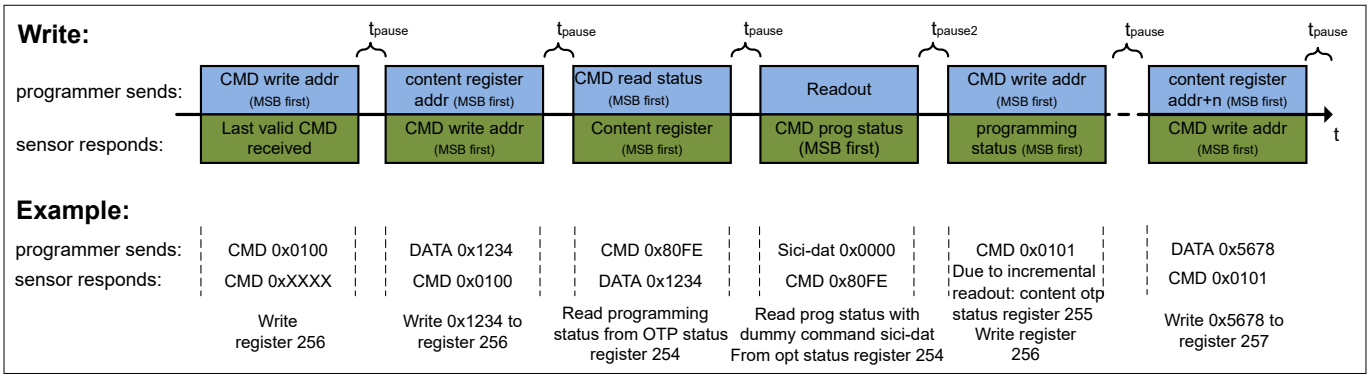

Figure 10 Frame order for write commands

Usually t_{pause} = 5 µs and t_{prog_pause} = 1 ms. The programmer should carefully follow those time gaps between commands and readouts.

2.2.3 Command and response frame

2.2.3.1 Command frame and CMD options

Table 2 Command frame encoding

Bits	Name	Description
17	CD	Fixed at 1 for command frame
16:15	CMD	Operation to be performed by interface: 00b...single/burst write (with automatic address increment) 01b...continuous write (without automatic address increment) 10b...single/burst read (with automatic address increment) 11b...extended commands for unlock, interface off and interface mode change.
14:1	ADDR	Register (start) address for read/ write command.
14:13	ECMD	Extended command: 00b..address resolution command 01b..interface mode change command 10b..unlock command
12:5	PROD_ID	Specific product ID: 0x0
4:1	S_ID	ID for lock/unlock command: 0x0
0	PAR	Odd parity over whole frame

The above Command frame encoding shows all possible bit structures. Usually the CMD ADDR and PAR is used. For special purposes, as unlocking or BIST, the extended command is needed. Examples in the User Manual will later illustrate the right procedures.

Frame encoding without extended command(prod_id, S_ID) when already in programming mode:

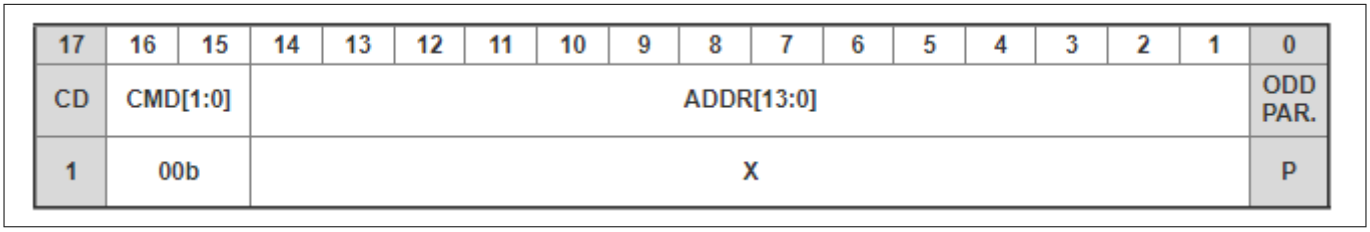

Figure 11

2.2.3.2 Unlock command

An unlock command includes a product and slave ID. When the SICI receives an unlock command, the received product ID is checked for 0xE0 and the slave ID is compared with the programmed ID in the CMTP register. In

case the IDs match, the SICI will transition to *active* state and thus unlock the decoding of normal read and write commands.

Different product and/or slave IDs on the other side force the SICI to interpret the command as a lock command, which will then transition to *silent* state. This allows to unlock one slave and lock another slave at the same time, which therefore prevents that two slaves are transmitting at the same time.

In case an unlock command with a wrong parity is received, the command is ignored. Therefore the master should check for a set status bit in the slave response of the next frame, which should also be visible if two slaves are driving the bus at the same time.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD	CMD[1:0]		ECMD[1:0]		PROD_ID[7:0]								S_ID[3:0]				ODD PAR.
1	11b		10b		X								Y				P

2.2.3.3 Write commands

The SICI interface will only react to write commands in *active* state.

Single/burst write command frame

This command performs an automatic address increment so that the next address can be written by simply sending another data frame. Otherwise, a new command interrupts the write command.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD	CMD[1:0]		ADDR[13:0]														ODD PAR.
1	00b		X														P

Continuous write command frame

This command performs no automatic address increment so that every data frame writes to the same address.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD	CMD[1:0]		ADDR[13:0]														ODD PAR.
1	01b		X														P

2.2.3.4 Read command

The SICI interface will only react to read commands in *active* state.

Single/burst read command frame

The SICI interface only implements a burst read command with automatic address increment. A continuous read-out can be emulated by resending the same read command during the response frame.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD	CMD[1:0]		ADDR[13:0]														ODD PAR.
1	10b		X														P

2.2.3.5 OFF command

The OFF command is always a broadcast command (hence the 1111b as broadcast ID) and every SICI slave on the bus will react to it in *silent* and *active* state and then transition to *off* state.

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CD	CMD[1:0]		ECMD[1:0]		CONST								Broadcast S_ID				ODD PAR.
1	11b		11b		00001111b								1111b				P

2.2.3.6 Response frame

Table 3 Response frame encoding

Bits	Name	Description
17	STAT	Status bit indicating error in previous command or frame.
16:1	DATA	Data for read command. Otherwise last content of SICI shift register
0	PAR	Odd parity over whole frame

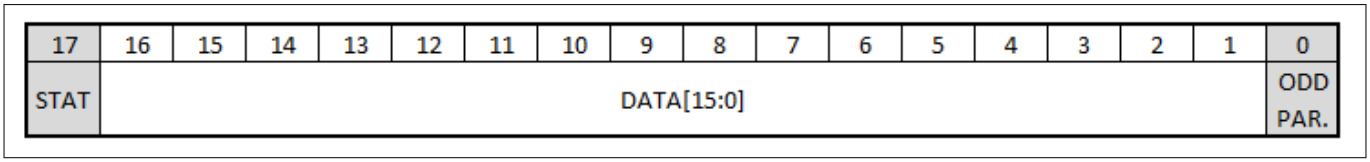

Figure 12

2.2.4 SICI Interface electrical and programming timing characteristics

Table 4

Parameter	Symbol	Min.	Typ.	Max.	Unit	Note or test condition
Programmer PWM period	T	40[1)]	-	120	µs	Determines interface transmission rate[2)]
Programmer low time to transmit “0”	t_{1_0}	0.28*T	0.33*T	0.38*T	µs	t_{2_0}=T - t_{1_0}[2)]
Programmer low time to transmit “1”	t_{1_1}	0.62*T	0.67*T	0.72*T	µs	t_{2_1}=T - t_{1_1}[2)]
Time difference between Programmer low and high	t_4	0.48*T	0.66*T	0.88*T	µs	$t_4 = 2 * \lvert t_{1_x} - t_{2_x} \rvert$[2)]
Programmer low pulse after PWM bit	t_3	0.07*T	-	0.2*T	µs	2)3)
Sensor response window	t_r	50	-	80	% t_4	
Interface reset time	t_{Res}	1.7	-	5	ms	2)4)
Input signal low level	$V_{low, in}$	-	-	0.3*V_{DD}	V	2)
Input signal high level	$V_{high, in}$	0.7*V_{DD}	-	-	V	2)
Output signal low level	$V_{low, out}$	-	-	0.2*V_{DD}	V	I_{sink} = 4mA
SICI line pull-up resistor	R_{pu}	1.45	-	2	kΩ	2)Limited due to SPC/SENT input
Pause between frames	t_{pause}	5	-	-	µs	

(table continues...)

Table 4 (continued)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Note or test condition
Time window for activation of programming interface	$t_{activation}$	-	-	4500	µs	time after power-up of the sensor to enter the programming mode with the command.
Duration of the CMTP programming pulse	t_{prog_pause}	1	-		ms	
Operating supply voltage	V_{DD}	4.4	5	6	V	2)
Programming supply voltage	V_{Prog}	15.5	-	18	V	2)

1) Achievable transmission rate (minimum Programmer PWM period) depends on parasitic capacities in external circuitry, 2 µs might be achievable only with no wiring and very low parasitic capacities.

2) Verified by design/characterization.

3) Rise/fall times due to parasitic capacitances on the line have to be added.

4) Maximum time for reset calculated for worst case interruption of SICI transmission.

2.3 Programming flow and procedure

2.3.1 Programming flow

Programming flow

The programming flow of the IC is visualized in figure 13. To program the CMTP registers, the sensor is brought into programming mode by sending the “Activate programming mode (SICI unlock, enabling testmode)” after power-on and t_activate. The desired address to be programmed is selected as described under Write commands(Section 2.2.3.3), written and programmed using the command sequence explained under programming procedure(Section 2.3.2). The procedure is repeated for all addresses in the CMTP area. Only after all CMTP register addresses are programmed, the firmware allows a SENT/SPC output. After programming, all the necessary CRCs need to be programmed and also the lock bit needs to be set if no further programming cycles are intended.

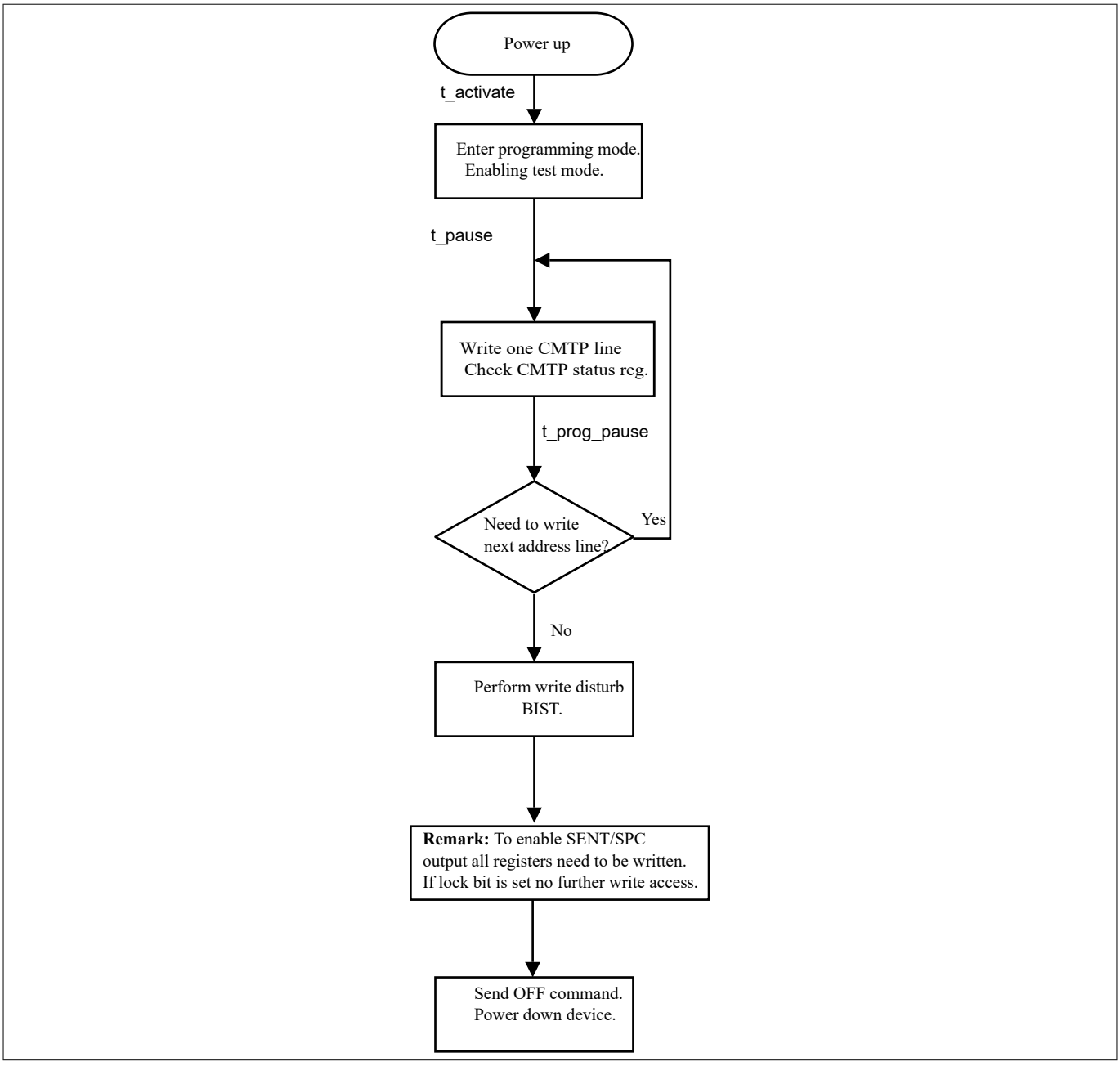

Figure 13 Programming flow

2.3.2 Programming procedure

Programming procedure

2 Programming

1. Instruction to enable test mode	Timing/level/cmd bits	Description
vdd	supply voltage	(typ. 5 V as defined in the data sheet)
Power Cycle		supply voltage off, wait 10 ms supply voltage on.
wait	t_activate	wait for supply settle
siciunlock	0xE000	product id, slave id (CMD bit, ECMD)
wait	t_pause	wait after unlock
sicicmd	0x00FD	write to unlock register
wait	t_pause	wait after cmd
sicidat	0x000F	otp diagnostic read and write enable
wait	t_pause	after wait sequence sensor is in test mode
2. programming sequence otp prog		%
vdd	V_PROG	add programming voltage as defined in data sheet: typ. 16 V
wait	t_activate	wait until V_PROG settle
sicicmd	0x0100	command for programming line 0
sicidat	0x1234	content line 0
wait	t_prog_pause	wait until line is programmed
sicicmd	0x80FE	read programming status from CMTP status registers 254 and 255
sicidat	0x0000	dummy command to readout register 254
sicicmd	0x0101	command for programming next line, sensor replies with auto-increment read of register 255
sicidat	0x5678	content line 1
wait	t_prog_pause	wait until line is programmed
sicicmd	0x80FE	read programming status from CMTP status registers 254 and 255
sicidat	0x0000	dummy command to readout register 254
sicicmd	0x0102	command for programming next line, sensor replies with auto-increment read of register 255
sicidat	0x9ABC	content line 2
wait	t_prog_pause	wait until line is programmed
sicicmd	0x80FE	read programming status from CMTP status registers 254 and 255
sicidat	0x0000	dummy command to readout register 254
sicicmd	0x0103	command for programming next line, sensor replies with auto-increment read of register 255
sicidat	0xCDEF	content line 3
wait	t_prog_pause	wait until line is programmed

Figure 14 Programming sequence

2.4 Register map

2.4.1 Definition

The IC embeds for some registers a multi-time programmable (CMTP) memory to store the configuration parameters.

The “C” in the acronym CMTP stands for customer programmable, intended is here the possibility that the customer of the chip can program some data, for example for special configuration or customization of the product to specific market or use-case. General in application programmability is not foreseen.

The “M” in C-MTP stands for multiple time programmable: Here is intended that a location can “apparently” be programmed more than once. In reality always an OTP (one time programmable) physical support is used and the multiple-programmability is achieved only by digital re-mapping of data, which of course limits the freedom and possibilities of re-programming.

The CMTP register has 60 registers and each register contains a 16-bit value. Registers can be either user-accessible or locked (define as "reserved"). The CMTP register is protected with multiple CRCs, when the configuration of a section is completed, its corresponding CRC needs to be updated accordingly.

When the entire configuration is performed, the CMTP register is set and cannot be re-written. This applies in general to all registers. The only exception applies to the registers in the address range from **273** to **306.** Those registers are redundantly available for a second programming cycle. The address 307 (block CRC) can be even written 5 times. Read access remains available for all registers.

The table below describes the register mapping of the IC. It contains a detailed description of each register.

Registers which contain values from the calculation paths are signed values which are treated in Two´s complement. Unsigned values or registers which have a specific bit coding are described in detail.

2.4.2 CMTP Register map

CMTP Register map

Register map of the CMTP register. Below registers are only writeable when the V_prog (programming voltage) is applied before the programming procedure.

Address	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
256	otp_chip_ID_1_reg	otp_chip_ID_1															
257	otp_chip_ID_2_reg	otp_chip_ID_2															
258	otp_chip_ID_3_reg	otp_chip_ID_3															
259	otp_chip_ID_4_reg	otp_chip_ID_4															
260	Reserved	Reserved															
261	Reserved																
262	Reserved																
263	Reserved																
264	Reserved																
265	Reserved																
266	Reserved																
267	Reserved																
268	Reserved																
269	Reserved																
270	Reserved																
271	Reserved																
272	otp_ifx_crc_reg	%								otp_ifx_crc							
273	otp_sent_spc_custom_reg	otp_sent_spc_stat		otp_spc_sent_data		otp_spc_sent_frame		otp_chip_id		otp_unit_time		otp_roll_cnt_en		otp_stat_crc_en	otp_var_trigger	otp_synch_mode_en	otp_pause_pulse_en
274	otp_sent_spc_slope_reg	Reserved												otp_sent_spc_mode	otp_sent_spc_slope		
275	otp_coil_cfg_reg	Reserved			otp_angle_2_direction	otp_angle_1_direction	otp_mech_inv_dir	otp_wings_adc2					otp_wings_adc1				
276	otp_ssm_custom1_reg	otp_ssm_custom1															
277	otp_ssm_custom2_reg	otp_ssm_custom2															
278	otp_ssm_custom3_reg	otp_ssm_custom3															
279	otp_vect_lim_reg	otp_vect_lim1_max								otp_vect_lim1_min							
280		otp_vect_lim2_max								otp_vect_lim2_min							
281	otp_offset_x1_reg	otp_offset_x1															
282	otp_offset_y1_reg	otp_offset_y1															
283	otp_offset_x2_reg	otp_offset_x2															
284	otp_offset_y2_reg	otp_offset_y2															
285	otp_adc_gain_reg	otp_y2_gain								otp_y1_gain							
286	otp_adc_orto_reg	otp_y2_orto								otp_y1_orto							
287	otp_angle_offset1_reg	otp_angle1_offset															
288	otp_angle_offset2_reg	otp_angle2_offset															
289	otp_torque_offset_reg	Reserved					otp_torque_offset										
290	otp_torque_stiffness_reg	Reserved					otp_torque_stiffness										
291	otp_lut_in_0_reg	otp_lut_0															
292	otp_lut_in_1_reg	otp_lut_1															
293	otp_lut_in_2_reg	otp_lut_2															
294	otp_lut_in_3_reg	otp_lut_3															
295	otp_lut_in_4_reg	otp_lut_4															
296	otp_lut_in_5_reg	otp_lut_5															
297	otp_lut_in_6_reg	otp_lut_6															
298	otp_lut_in_7_reg	otp_lut_7															
299	otp_lut_in_8_reg	otp_lut_8															
300	otp_lut_in_9_reg	otp_lut_9															
301	otp_lut_in_10_reg	otp_lut_10															
302	otp_lut_in_11_reg	otp_lut_11															
303	otp_lut_in_12_reg	otp_lut_12															
304	otp_lut_in_13_reg	otp_lut_13															
305	otp_lut_in_14_reg	otp_lut_14															
306	otp_lut_in_15_reg	otp_lut_15															
307	otp_calibration_crc_reg	Reserved							otp_lock	otp_calibration_crc							

Figure 15 CMTP Register map

2.4.3 CMTP Register description

CMTP Register description - part 1

Address/ position	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
256	otp_chip_ID_1_reg		otp_chip_ID_1															
<15:0>	otp_chip_ID_1	Chip ID 1																
257	otp_chip_ID_2_reg		otp_chip_ID_2															
<15:0>	otp_chip_ID_2	Chip ID 2																
258	otp_chip_ID_3_reg		otp_chip_ID_3															
<15:0>	otp_chip_ID_3	Chip ID 3																
259	otp_chip_ID_4_reg		otp_chip_ID_4															
<15:0>	otp_chip_ID_4	Chip ID 4																
272	otp_ifx_crc_reg		otp_unused_272								otp_ifx_crc							
<7:0>	otp_ifx_crc	block CRC for Infineon area																
<15:8>	otp_unused_272	empty																
273	otp_sent_spc_custom_reg		otp_sent_spc_stat		otp_spc_sent_data		otp_spc_sent_frame		otp_chip_id		otp_unit_time			otp_roll_cnt_en	otp_stat_crc_en	otp_var_trigger	otp_synch_mode_en	otp_pause_pulse_en
0	otp_pause_pulse_en	Pause pulse enable bit 0: no pause pulse 1: pause pulse enabled (default)																
1	otp_synch_mode_en	synch mode configuration SPC protocol 0: bus mode 1: synch mode (default)																
2	otp_var_trigger	variable Trigger for SPC 0: no varable trigger (default) 1: variable trigger enabled																
3	otp_stat_crc_en	includes status nibble in crc 0: status nibble not included 1: status nibble included (default)																
4	otp_roll_cnt_en	rolling counter enable 0: rolling counter disabled 1: rolling counter enabled																
<7:5>	otp_unit_time	value for unit time configuration 000: 1 µs 001: 1.5 µs 001: 2 µs 011: 2.5 µs 100: 3 µs																
<9:8>	otp_chip_id	SPC ID																
<12:10>	otp_spc_sent_frame	defines the frame lenght of the sent/spc protocol 000: protocol A 001: protocol B 010: protocol C %																
<14:13>	otp_spc_sent_data	defines the data of the spc sent protocol 00 :torque; angle1 / angle2 alternating 01: torque; angle 1 10: torque; angle 2																
15	otp_sent_spc_stat	defines the status nibble for sent or spc 0: SENT status nibble 1: SPC status nibble																
274	otp_sent_spc_slope_reg		otp_unused_274												otp_sent_spc_mode	otp_sent_spc_slope		
<2:0>	otp_sent_spc_slope	sent_spc_slope setting 0: 3 µs 001: 2.5 µs 010: 2 µs 011: 1.5 µs 100: 1 µs 101: 1 µs 110: 1 µs 111: slope disabled																
3	otp_sent_spc_mode	0: spc_mode 1: sent_mode																
<15:4>	otp_unused_275	empty																

Figure 16 CMTP Registers with description - part 1

Remarks to figure 16:

The otp_sent_spc_stat in register 273 (bit 15) only configures the status nibble used for the status message. It is possible to configure a SENT status nibble even when SPC interface configuration is enabled. Same applies when SENT is configured for the SPC status nibble.

TLE480x – FE chip-ID coding scheme

Reg # \ Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
256	lot-serial-number										not used				fab-location	
257	lot-date										wafer-number					
258	x-coordinate										y-coordinate					

Example: ZA319013-08 48/111

- fab-location: ZA (equivalent to 3_{dec})
- lot-date: 319 (year + cw)
- lot-serial-number: 013
- wafer-number: 08
- x-coordinate: 48
- y-coordinate: 111

- $Reg256_{dec}$: 835
- $Reg257_{dec}$: 20424
- $Reg258_{dec}$: 12399

Calculation:

- fab-location = $Reg256_{dec}$ mod(2^6)
- lot-serial-number = ($Reg256_{dec}$ – fab-location) / 2^6

- wafer-number = $Reg257_{dec}$ mod(2^6)
- lot-date = ($Reg257_{dec}$ – wafer-number) / 2^6

- y-coordinate = $Reg258_{dec}$ mod(2^8)
- x-coordinate = ($Reg258_{dec}$ – y-coordinate) / 2^8

Figure 17 Chip ID coding scheme

CMTP Register description - part 2

275	otp_coil_cfg_reg		otp_unused_275	otp_angle_2_direction	otp_angle_1_direction	otp_mech_inv_dir	otp_wings_adc2	otp_wings_adc1
<4:0>	otp_wings_adc1	number of target wings for ADC1. It is also possible to program as wing number the greatest common divider: e.g: wing1 = 6,wing2 = 9 could be programmed as wing1 = 2 and wing2 = 3						
<9:5>	otp_wings_adc2	number of target wings for ADC2						
10	otp_mech_inv_dir	inversion of mechanical angle						
11	otp_angle_1_direction	changes the direction of angle 1 value						
12	otp_angle_2_direction	changes the direction of angle 2 value						
<15:13>	reserved	%						
276	otp_ssm_custom1_reg		otp_ssm_custom1					
<15:0>	otp_ssm_custom1	custom specific Short serial message data						
277	otp_ssm_custom2_reg		otp_ssm_custom2					
<15:0>	otp_ssm_custom2	custom specific Short serial message data						
278	otp_ssm_custom3_reg		otp_ssm_custom3					
<15:0>	otp_ssm_custom3	custom specific Short serial message data						
279	otp_vect_lim1_reg		otp_vect_lim1_max					otp_vect_lim1_min
<7:0>	otp_vect_lim_min	minimum Limit for vector length error						
<15:8>	otp_vect_lim_max	maximum Limit for vector length error						
280	otp_vect_lim2_reg		otp_vect_lim2_max					otp_vect_lim2_min
<7:0>	otp_vect_lim_min	minimum Limit for vector length error						
<15:8>	otp_vect_lim_max	maximum Limit for vector length error						
281	otp_offset_x1_reg		otp_offset_x1					
<12:0>	otp_offset_x1	offset compensation value for X1						
<15:13>	otp_unused_280	empty						
282	otp_offset_y1_reg		otp_offset_y1					
<12:0>	otp_offset_y1	offset compensation value for Y1						
<15:13>	otp_unused_281	empty						
283	otp_offset_x2_reg		otp_offset_x2					
<12:0>	otp_offset_x2	offset compensation value for X2						
<15:13>	otp_unused_282	empty						
284	otp_offset_y2_reg		otp_offset_y2					
<12:0>	otp_offset_y2	offset compensation value for Y2						
<15:13>	otp_unused_283	empty						
285	otp_adc_gain_reg		otp_y2_gain					otp_y1_gain
<7:0>	otp_y1_gain	gain for adc1 compensation						
<15:8>	otp_y2_gain	gain for adc2 compensation						
286	otp_adc_orto_reg		otp_y2_orto					otp_y1_orto
<7:0>	otp_y1_orto	orthogonality for adc1 compensation						
<15:8>	otp_y2_orto	orthogonality for adc2 compensation						
287	otp_angle_offset1_reg		otp_angle1_offset					
<15:0>	otp_angle1_offset	offset for angle1 calibration						
288	otp_angle_offset2_reg		otp_angle2_offset					
<15:0>	otp_angle2_offset	offset for angle2 calibration						
289	otp_torque_offset_reg		otp_unused_289					otp_torque_offset
<11:0>	otp_torque_offset	torque offset value						
<15:12>	otp_unused_288	empty						
290	otp_torque_stiffness_reg		otp_unused_290					otp_torque_stiffness
<11:0>	otp_torque_stiffness	torque_stiffness						
<15:12>	otp_unused_289	empty						
291-298	otp_lut_in_1_reg to otp_lut_in_8_reg		otp_lut_0 to otp_lut_7					
<15:0>	otp_lut	multi point calibration input value 0 - 7						
299-306	otp_lut_out_1_reg to otp_lut_out_8_reg		otp_lut_8 to otp_lut_15					
<15:0>	otp_lut	multi point calibration output value 8 - 15						
307	otp_calibration_crc_reg		otp_unused_307				otp_lock	otp_calibration_crc
<7:0>	otp_calibration_crc	block CRC for calibration area 273-306						
<15:8>	otp_unused_290	empty						

Figure 18 CMTP register with descriptions - part 2

Remarks to the figure 18:

CRC register **otp_calibration_crc** can be written 5 times if no **otp_lock** bit has been set. If **otp_lock** is set after first write cycle no further write cycles are possible. When setting the **otp_lock** bit it is important to write also the right CRC value in the same register 307 in the same write cycle.

Writing to the otp_unused areas is not recommended: This would cause a wrong CRC check. The firmware assumes the default values in the otp_unused areas and won't read into the shadow registers newly written values.

Footnote to parameters:

Parameter definitions which have not been described in detail here can be found under chapter: Parameter mapping and Calibration(section 3 about calibration and section 5 about parameter mapping)

2.4.4 Working registers

The working registers below give the possibility to read out via SICI raw values and processed values to check whether the configuration written into the CMTP or into the CMTP read out register for simulation purposes perform as intended.

Address/ position	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	angle1_dsp_int_reg		angle1_dsp_int															
<15:0>	angle1_dsp_int	angle1 value calculated by interface dsp																
5	angle2_dsp_int_reg		angle2_dsp_int															
<15:0>	angle2_dsp_int	angle2 value calculated by interface dsp																
6	vector_len_dsp_adc1_int_reg		vector_len_adc1_dsp_int															
<15:0>	vector_len_adc1_dsp_int	Vector length of ADC1 calculated by interface dsp [unsigned]																
7	vector_len_adc2_dsp_int_reg		vector_len_adc2_dsp_int															
<15:0>	vector_len_adc2_dsp_int	Vector length of ADC2 calculated by interface dsp [unsigned]																
8	torque_dsp_int_reg		torque_dsp_int															
<15:0>	torque_dsp_int	torque value calculated by interface dsp																
9	offset_cal_x1_dsp_int_reg		offset_cal_x1_dsp_int															
<15:0>	offset_cal_x1_dsp_int	offset for ADC1 X compensation																
10	offset_cal_y1_dsp_int_reg		offset_cal_y1_dsp_int															
<15:0>	offset_cal_y1_dsp_int	offset for ADC1 Y compensation																
11	offset_cal_x2_dsp_int_reg		offset_cal_x2_dsp_int															
<15:0>	offset_cal_x2_dsp_int	offset for ADC2 X compensation																
12	offset_cal_y2_dsp_int_reg		offset_cal_y2_dsp_int															
<15:0>	offset_cal_y2_dsp_int	offset for ADC2 Y compensation																
13	mech_angle_dsp_int_reg		mech_angle_dsp_int															
<15:0>	mech_angle_dsp_int	mechanical Angle calculated by interface DSP																
14	x1_o_dsp_int_reg		x1_o_dsp_int															
<15:0>	x1_o_dsp_int	offset compensated adc1 x value																
15	y1_f_dsp_int_reg		y1_f_dsp_int															
<15:0>	y1_f_dsp_int	offset and gain compensated adc1 y value																
16	angle1_mc_dsp_int_reg		angle1_mc_dsp_int															
<15:0>	angle1_mc_dsp_int	multi point compensated angle 1 value																
17	x2_o_dsp_int_reg		x2_o_dsp_int															
<15:0>	x2_o_dsp_int	offset compensated adc2 x value																
18	y2_f_dsp_int_reg		y2_f_dsp_int															
<15:0>	y2_f_dsp_int	offset and gain compensated adc2 y value																
19	angle2_mc_dsp_int_reg		angle2_mc_dsp_int															
<15:0>	angle2_mc_dsp_int	multi point compensated angle 2 value																
20	diff_angle_dsp_int_reg		diff_angle_dsp_int															
<15:0>	diff_angle_dsp_int	differential angle value angle																
21	angle1_dsp_saf_reg		angle1_dsp_saf															
<15:0>	angle1_dsp_saf	angle1 value calculated by safety dsp																
22	angle2_dsp_saf_reg		angle2_dsp_saf															
<15:0>	angle2_dsp_saf	angle2 value calculated by safety dsp																
23	torque_saf_reg		torque_dsp_saf															
<15:0>	torque_dsp_saf	torque value calculated by safety dsp																

Figure 19 Working registers

Remark to figure 19:

For the vector length registers **6 and 7** the sensitivity settings for ADC1 and ADC2 apply:

Small range(Range 0):

$$\frac{28.6 \quad mVp}{2^{16} \quad LSB16} \tag{1}$$

Medium range(Range 1):

$$\frac{57.2 \quad mVp}{2^{16} \quad LSB16} \tag{2}$$

Working registers part 2

Registers which hold initially the content from the CMTP:

Address	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
118	sent_spc_custom_reg		sent_spc_stat	spc_sent_data		spc_sent_frame			chip_id		unit_time			roll_cnt_en	stat_crc_en	var_trigger	synch_mode_en	pause_pulse_en
0	pause_pulse_en	Pause pulse enable bit 0: No pause pulse 1: Pause pulse enabled (default)																
1	synch_mode_en	synch mode configuration SPC protocol 0: bus mode 1: synch mode (default)																
2	var_trigger	Variable Trigger for SPC 0: No varable trigger (default) 1: variable trigger enabled																
3	stat_crc_en	includes status nibble in crc																
4	roll_cnt_en	Rolling counter enable 0: status nibble not included 1: status nibble included (default)																
<7:5>	unit_time	value for unit time configuration 000: 1us 001: 1.5us 010: 2us 011: 2.5us 100: 3us																
<9:8>	chip_id	SPC ID																
<12:10>	spc_sent_frame	defines the frame lenght of the sent/spc protocol 000: protocol A 001: protocol B 010: protocol C %																
<14:13>	spc_sent_data	defines the data of the spc sent protocol 00 :Torque; Angle1 / Angle2 alternating 01: Tourque; Angle 1 10: Tourque; Angle 2 %																
15	sent_spc_stat	defines status nibble if sent or spc 0: SENT status nibble 1: SPC status nibble																
119	sent_spc_slope_reg														sent_c_mode	sent_spc_slope		
<2:0>	sent_spc_slope	sent_spc_slope setting 0: 3us 1: 2,5us 2: 2us 3: 1,5us 4: 1us 5: 1us 6: 1us 7: slope disabled																
3	sent_c_mode	customer sent enable mode (just enabled when isent pritio is switched to customer mode 0: spc_mode 1: sent_mode																
120	coil_cfg_reg					angle_2_direction	angle_1_direction	mech_inv_dir	wings_adc2					wings_adc1				
<4:0>	wings_adc1	Number of poles for ADC1																
<9:5>	wings_adc2	Number of poles for ADC2																
10	mech_inv_dir	Inversion of mechanical angle																
11	angle_1_direction	changes the direction of angle 1 value																
12	angle_2_direction	changes the direction of angle 2 value																
13	NA	%																
121	ssm_custom1_reg		ssm_custom1															
<15:0>	ssm_custom1	short serial message customer register 1																
122	ssm_custom2_reg		ssm_custom2															
<15:0>	ssm_custom2	short serial message customer register 2																
123	ssm_custom3_reg		ssm_custom3															
<15:0>	ssm_custom3	short serial message customer register 3																
124	vect_lim1_reg		vect_lim1_max											vect_lim1_min				
<7:0>	vect_lim1_min	minimum Limit for vector length error rotor 1																
<15:8>	vect_lim1_max	maximum Limit for vector length error rotor 1																
125	vect_lim2_reg		vect_lim2_max											vect_lim2_min				
<7:0>	vect_lim2_min	minimum Limit for vector length error rotor 2																
<15:8>	vect_lim2_max	maximum Limit for vector length error rotor 2																
126	offset_x1_reg		offset_x1															
<15:0>	offset_x1	offset compensation value for X1																
127	offset_y1_reg		offset_y1															
<15:0>	offset_y1	offset compensation value for Y1																
128	offset_x2_reg		offset_x2															
<15:0>	offset_x2	offset compensation value for X2																
129	offset_y2_reg		offset_y2															
<15:0>	offset_y2	offset compensation value for Y2																
130	adc_gain_reg		y2_gain											y1_gain				
<7:0>	y1_gain	gain for adc1 compensation																
<15:8>	y2_gain	gain for adc2 compensation																
131	adc_orto_reg		y2_orto											y1_orto				
<7:0>	y1_orto	Ortogonality for adc1 compensation																
<15:8>	y2_orto	Ortogonality for adc2 compensation																
132	angle_offset1_reg		angle1_offset															
<15:0>	angle1_offset	Offset for Angle1 calibration																
133	angle_offset2_reg		angle2_offset															
<15:0>	angle2_offset	Offset for Angle2 calibration																
134	torque_offset_reg						torque_offset											
<11:0>	torque_offset	Torque offset value																
135	torque_stiffness_reg						torque_stiffness											
<11:0>	torque_stiffness	Torque_stiffness																

Figure 20 Read in registers

Remarks to the figure:

The registers in the figure before can be programmed via SICI. When writing to those registers only the normal 5 V voltage level is applied. This way it is possible to fine tune values and to read then via SICI the raw values or angular digital values through the working registers in figure 20. After reset in contrast to the CMTP section, the values programmed here through SICI are again filled with the CMTP values.

The firmware reads values from these working registers every 156 µs.

Further remarks to configuration settings which are aligned to each other:

Slope time and unit time configuration should be aligned in the same manner to obtain best communication results: It is recommended to use for a unit_time of 1 µs also for the sent_spc_slope the 1 µs slope rise/fall time. It is recommended that UT time equals slope time. A slow slope could result in a bad signal quality and a too fast slope in comparison to UT could result in bad EMC behaviour.

2.4.5 CMTP Status Register

CMTP status registers

After each programmed line, it is necessary to read out the OTP status registers to check whether error flags are raised or access to the OTP was granted. In the example programming procedure (Figure 13), both registers are read to determine the programming success. Here with the read command **0x80FE** for register **254** an incremental read is started. **254** is read with the dummy command **0x0000**. **255** is read with the next write command in parallel.

It is also possible to check how many times certain registers, which support multiple programming cycles, have been written and if further programming cycles are possible. To perform this check, it is crucial to perform a read to determine the remaining programming cycles. Afterwards with another read-command **254 (0x80FF)** and a command (to read the content) it is possible to determine the value in the bit area <12:8>. The value shows if 2, 1 or 0 programming cycles are still possible.

Address/ position	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
254	otp_stat_reg		read_access	otp_vdd2v5_2	otp_vdd2v5_1	otp_program_access					otp_access_granted	otp_cell_virgin	otp_ecc_stat		search_valid	otp_data_valid	otp_ahb_err	otp_busy
0	otp_busy	Status of the otp 0: cmtp_ok 1: cmtp_busy																
1	otp_ahb_err	Signalization if ahb bus returned an error(advanced high performance bus) 0: otp_ok 1: otp_ahb_err																
2	otp_data_valid	Search valid indication 0: otp_data_not_valid 1: otp_data_valid																
3	search_valid	Search valid indication 0: otp_data_not_valid 1: otp_data_valid																
<5:4>	otp_ecc_stat	0: no_error 1: one_error 2: two_errors 3: three_errors																
6	otp_cell_virgin	Indication if the cell got programmed 0: cell_programmed 1: cell_virgin																
7	otp_access_granted	Indication if access is denied 0: access_denied 1: access_granted																
<12:8>	otp_program_access	still available further program access																
13	otp_vdd2v5_1	voltage vdd2v5 in ok																
14	otp_vdd2v5_2	voltage vdd2v5 out ok																
15	read_access	Read acces of OTP 0: NOT a READ access 1: Read access																
255	otp_stat_2_reg							locked	vddprog_2	vddprog_1	seq_code					lock_fail_n	out_of_attempt_n	data_check
0	data_check	Indication that embedded data verification was successful 1: pass 0: fail																
1	out_of_attempt_n	Programming failed due to already programmed cell 1: Programming possible 0: Out of attempts, programming is fail																
2	lock_fail_n	Indication of impossibility to programm due to write protection 1: programming was possible 0: cmtp write protected, program not possible																
<7:3>	seq_code	Sequence code: 01000 after successful programming																
8	vddprog_1	Status programming voltage 0: invalid voltage 1: valid voltage																
9	vddprog_2	Status programming voltage 0: invalid voltage 1: valid voltage																
10	locked	Status of the otp 0: cmtp_unlocked 1: cmtp_locked																

Figure 21 CMTP status register

2.4.6 Hardware status register

Hardware status register

The following hardware status registers enable the possibility to determine via SICI why via SENT an error code is sent. The error flag indicates the triggered monitoring or the root cause.

Address	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	stat_hw_0_reg		stat_ssm_prot_err	stat_err_code				stat_rxy2_vcm	stat_rxy1_vcm	stat_tx_vcm	stat_ovuv2d	stat_ovuv1d	stat_ovuv2a	stat_ovuv1a	stat_out_ov	stat_vddp_ov	stat_vddp_uv	stat_rst
0	stat_rst	Power-on hardware reset indication for interface.																
1	stat_vddp_uv	Undervoltage on supply.																
2	stat_vddp_ov	Overvoltage on supply.																
3	stat_out_ov	Overvoltage indication of out pad																
4	stat_ovuv1a	Voltage comparator 1a																
5	stat_ovuv2a	Voltage comparator 2a																
6	stat_ovuv1d	Voltage comparator 1d																
7	stat_ovuv2d	Voltage comparator 2d																
8	stat_tx_vcm	TX open short error																
9	stat_rxy1_vcm	RXY1 open short error																
10	stat_rxy2_vcm	RXY2 open short error																
<14:12>	stat_err_code	Error code on protocol 000: no error 001: Supply/temperature monitoring, internal error monitoring (SENT/SPC error code 12bit:4089) 010: Transceiver coils monitoring(open,short and resistance measurement), Transceiver coils external parameters monitoring(frequency) (SENT/SPC error code 12bit:4090) 011: Transceiver coils external parameters monitoring(amplitude measurement) (SENT/SPC error code 12bit:4091) 100: Receiver coils monitoring(open, short, resistance measurement) (SENT/SPC error code 12bit: 4092) 101: Receiver coils contact monitoring(vector length) (SENT/SPC error code 12bit:4093) 110: ADC/angle/torque crosscheck (SENT/SPC error code 12bit:4094) %																
15	stat_ssm_prot_err	Gives the status of the error flag on protocol																
1	stat_hw_1_reg								stat_rom_su_dsp_saf	stat_rom_su_dsp_int	stat_rom_run_dsp_saf	stat_rom_run_dsp_int	stat_sched_dsp_saf	stat_sched_dsp_int	stat_cordic_overflow	stat_cmtp_rd	stat_cmtp_ecc	stat_cmtp_prg
0	stat_cmtp_prg	CMTP not fully programmed																
1	stat_cmtp_ecc	CMTP ECC error																
2	stat_cmtp_rd	CMTP readout error																
3	stat_cordic_overflow	Cordic overflow error																
4	stat_sched_dsp_int	Scheduling error of interface dsp																
5	stat_sched_dsp_saf	Scheduling error of safety dsp																
6	stat_rom_run_dsp_int	DSP INT EDC runtime error																
7	stat_rom_run_dsp_saf	DSP SAF EDC runtime error																
8	stat_rom_su_dsp_int	DSP INT EDC startup error																
9	stat_rom_su_dsp_saf	DSP SAF EDC startup error																
2	stat_fw_0_reg				stat_isma_bist	stat_torque_cmp	stat_adc_cmp	stat_rx_amp2	stat_rx_amp1	stat_tx_amp_cross	stat_tx_amp	stat_angle_cmp2	stat_angle_cmp1	stat_crc	stat_block_crc_usr	stat_block_crc_ifx	stat_tx_frq	stat_t_lin_ot_shdn
0	stat_t_lin_ot_shdn	Overtemperature shutdown																
1	stat_tx_frq	Error on LC frequency																
2	stat_block_crc_ifx	Block CRC error on IFX area																
3	stat_block_crc_usr	Block CRC error on user area																
4	stat_crc	Security key register CRC check																
5	stat_angle_cmp1	angle comparison of redundant angle 1																
6	stat_angle_cmp2	angle comparison of redundant angle 2																
7	stat_tx_amp	Ic amplitude error																
8	stat_tx_amp_cross	Crosscheck of TX amplitude with value of GP_ADC2																
9	stat_rx_amp1	Checks the vector lengths of ADC1																
10	stat_rx_amp2	Checks the vector lengths of ADC2																
11	stat_adc_cmp	ADC value comparison																
12	stat_torque_cmp	Torque comparison																
13	stat_isma_bist	ISMA bist error																
3	stat_fw_1_reg													stat_ltx	stat_lry2	stat_lrx2	stat_lry1	stat_lrx1
0	stat_lrx1	Resistance error of coil1 X																
1	stat_lry1	Resistance error of coil1 Y																
2	stat_lrx2	Resistance error of coil2 X																
3	stat_lry2	Resistance error of coil2 Y																
4	stat_ltx	Resistance error of coil TX																
166	hw_fw_id_reg										hw_id				fw_id			
<0:3>	fw_id	Firmware ID																
<5:8>	hw_id	Hardware ID																

Figure 22 Hardware status register overview with descriptions

Clearing of status flags with SICI test interface:

- The status flags contained in the registers stat_hw_0_reg, stat_hw_1_reg, stat_fw_0_reg, stat_fw_1_reg are cleared at each read out with exception of the flag stat_ssm_prot_err. At each single register read out, only the flags contained in that register are cleared.
- The flag stat_ssm_prot_err contains the information if one of the error flags in the four register is still active and this flag is cleared only if all other flags are also cleared. It may be necessary to read out all four registers to clear this flag after a single error activation.
- Hardware safety flags have a fast update rate. At persistent errors they go nearly immediately to the set state after clearing.
- Firmware safety flags have an update rate of <=312 µs so at fast read out it is possible to read back a cleared state even at persistent errors.
- If using read out with autoincrement the register n+1 will be automatically cleared after readout of register n.
  - To prevent auto clearing, no dummy data should be sent, instead the next command should be sent
- It is recommended to read out all 4 stat-registers to recalculate all error flags and to remove old error flags.

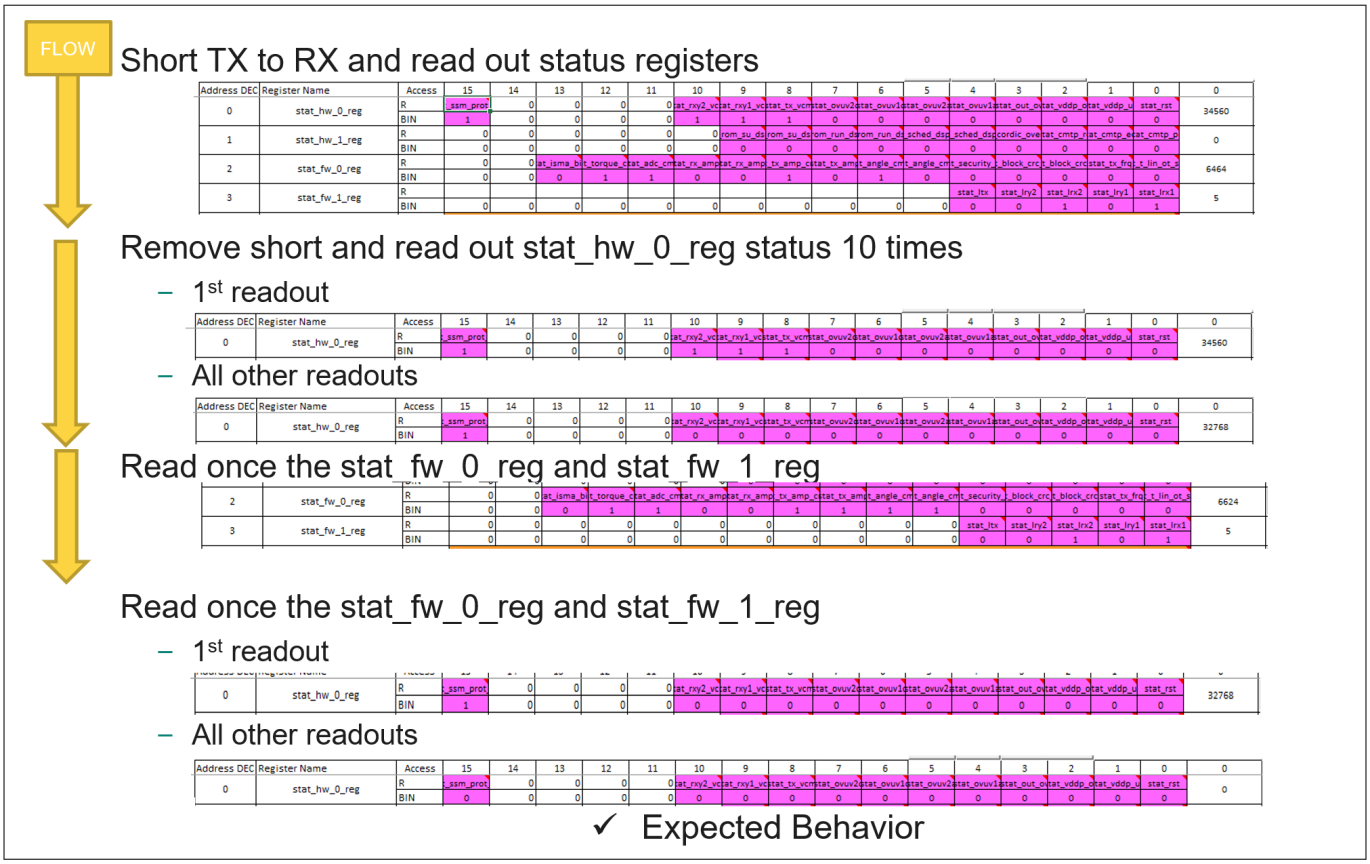

Figure 23 Flag clearing flow

Clearing of safety flags in SENT/SPC:

- All the safety flags are cleared at each SENT/SPC protocol.
- Persistent errors will show persistent SENT/SPC protocols containing error information if the protocol update rate is >312 µs.
- At faster protocol update rates than 312 µs it may be that not each protocol contains the error information.

2.4.7 ADC status registers overview

Address	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
36	adc1_x_reg		adc1_x															
<15:0>	adc1_x	ADC1 X channel																
37	adc1_y_reg		adc1_y															
<15:0>	adc1_y	ADC1 Y channel																
38	adc2_x_reg		adc2_x															
<15:0>	adc2_x	ADC2 X channel																
39	adc2_y_reg		adc2_y															
<15:0>	adc2_y	ADC2 Y channel																
40	gp_adc1_osc_amp_raw_reg		gp_adc1_osc_amp															
<15:0>	gp_adc1_osc_amp	lc oscillator amplitude from general purpose ADC1																
41	gp_adc1_x_reg		gp_adc1_x															
<15:0>	gp_adc1_x	X ADC reg from general purpose ADC1																
42	gp_adc1_x_res_reg		gp_adc1_x_res															
<15:0>	gp_adc1_x_res	X ADC reg from general purpose ADC1																
43	gp_adc1_y_reg		gp_adc1_y															
<15:0>	gp_adc1_y	Y ADC reg from general purpose ADC1																
44	gp_adc1_y_res_reg		gp_adc1_y_res															
<15:0>	gp_adc1_y_res	Y ADC reg from general purpose ADC1																
45	gp_adc1_temp_reg		gp_adc1_temp															
<15:0>	gp_adc1_temp	Y ADC reg from general purpose ADC1																
46	gp_adc2_osc_amp_raw_reg		gp_adc2_osc_amp															
<15:0>	gp_adc2_osc_amp	lc oscilator amplitude from general purpose ADC2																
47	gp_adc2_x_reg		gp_adc2_x															
<15:0>	gp_adc2_x	X ADC reg from general purpose ADC2																
48	gp_adc2_x_res_reg		gp_adc2_x_res															
<15:0>	gp_adc2_x_res	X ADC reg from general purpose ADC2																
49	gp_adc2_y_reg		gp_adc2_y															
<15:0>	gp_adc2_y	Y ADC reg from general purpose ADC2																
50	gp_adc2_y_res_reg		gp_adc2_y_res															
<15:0>	gp_adc2_y_res	Y ADC reg from general purpose ADC2																
51	gp_adc2_temp_reg		gp_adc2_temp															
<15:0>	gp_adc2_temp	Y ADC reg from general purpose ADC2																
52	gp_adc2_lc_res_reg		gp_adc2_lc_res															
<15:0>	gp_adc2_lc_res	Resistor value of LC measured by ADC2																
53	osc_amp_reg			adc2_range		adc1_range			osc_amplitude									
<11:0>	osc_amplitude	bias current for lc oscillator																
<13:12>	adc1_range	Manual range seting for adc1 range configuration 00: small range (small signals) 01: medium range (medium signals) 10: high range (not used) 11: high range (not used)																
<15:14>	adc2_range	Manual range seting for adc2 range configuration 00: small range (small signals) 01: medium range (medium signals) 10: high range (not used) 11: high range (not used)																
54	lc_frequency_reg			operating_mode					lc_frequency									
<10:0>	lc_frequency	integration values of the demodulation signal over 100 µs 4 MHz oscillation signal gives a value of 400																
<15:14>	operating_mode	operating modes 0: om_reset 1: om_startup 2: om_running 3: om_test																

Figure 24 ADC read only working register

Remarks to figure:

osc_amp_reg:

- adc1 and adc2 ranges:
  - After a read to this register only the states "00" small range and "01" medium range are possible.

adc1/2_x/y_reg and gp_adc1/2_x/y_reg:

small range:

$$Amplitude \quad peak \quad to \quad peak \quad = \quad typ. \quad 57.2 \quad mVpp$$

$$\frac{28.6 \quad mVp}{2^{15}} = \quad 0.87 \times 10^{-3} \frac{mV}{LSB16} \tag{3}$$

medium range:

$$Amplitude\ \ peak\ \ to\ \ peak\ \ =\ \ typ.\ \ 114.2\ \ mVpp$$
$$\frac{57.2\ \ mVp}{2^{15}}\ \ =\ \ 1.75\times10^{-3}\ \frac{mV}{LSB16} \tag{4}$$

osc_amp_reg:

osc_amplitude: To read out the bias current for the LC Oscillator.

$$typ.\ \ 7.5\ \ mA\ \ \rightarrow\frac{7.5\ \ mA}{2^{12}}\ \ =\ \ ->1.83*10^{-3}\ \ \frac{mA}{LSB12} \tag{5}$$

gp_adc1/2_osc_amp_reg:

$$typ.\ \ 1.2\ \ Vp\rightarrow\ \ typ.\ \ 16650\ \ LSB16 \tag{6}$$

The LC circuit consists of the TX coil, the TX capacitors and the TX_A and TX_B connecting pins. The amplitude of typ. 1.2Vp (typ. 2.4Vpp) is controlled through the gp_adc1.

gp_adc1/2_x/y/lc_res_reg:

For the TX-coil/RX-coils the following sensitivities can be applied to determine the resistor values:

TX:

$$typ.\ \ \ \ 4.36\frac{LSB16}{ohm} \tag{7}$$

RX:

$$typ.\ \ \frac{43.6\ \ LSB16}{ohm} \tag{8}$$

gp_adc1/2_temp_reg:
The temperature registers have a typical offset of -10570 LSB16 to 0°C (0°C corresponds typically to -10570 LSB16)

- The typical sensitivity: 128 LSB16/°C

gp_adc1/2_temp_reg:
It is important to distinguish between the gp_adc1/2 and the adc1/2 when operating in the small signal range. When comparing the GP_ADC1/2 values with the ADC1/2 value, the ADC1/2 resolution is 2 times higher.

2.4.8 Temperature register ADC

Temperature register ADC

Address	Register Name	Description	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
78	t_lin_dsp_int_reg		t_lin_dsp_int															
<15:0>	t_lin_dsp_int	DSP INT channel calibrated local temperature.																
79	t_lin_dsp_saf_reg		t_lin_dsp_saf															
<15:0>	t_lin_dsp_saf	DSP SAF channel calibrated local temperature.																

Figure 25 Calibrated temperature registers

Remarks to the figure:

The register contains the calibrated temperature from redundant sensors(ADC1/2,GP_ADC1/2).

- The 16 bit signed integer value of 78 and 79 has the typical value range for -40°...175°C.
- The actual value is between -1280 ... 5600.
  - -40° = typ. 1280LSB16
  - 175° = typ. 5600LSB16

Sensitivity: **32LSB16/°C**

2.5 Built in self test - BIST

Write-Disturb BIST - WRDBIST

Instruction	Timing/level/cmd bits	Description
vdd	supply voltage	(typ. 5V as defined in the data sheet)
wait	t_activate	wait until supply voltage settle
siciunlock	0xE000	product id, slave id (CMD bit, ECMD)
wait	t_pause	
next step	TEMP_REG	temp_reg sequence
sicicmd	0x0150	write the temp-register
wait	t_pause	
sicidat	0x0000	
next step	CMTP_CFG	cmtp configuration sequence
sicicmd	0x10FD	single write : address 253
sicidat	0x0011	write c_mtp config register
wait	t_pause	
next step	WRDBIST	WRDBIST sequence
sicicmd	0x8145	WRDBIST (read cmd)
wait	t_prog_pause	
sicidat	0x0000	Dummy command for sici results
wait	2 x t_prog_pause	
Result when sending "sicidat (0x0000)" should be: 0xC26F		

Figure 26 WRDBIST sequence

If the result of the WRDBIST test differs from the expected **0xC26F,** check that all steps have been performed as described. When the discrepancy persists, try the procedure again with a different device.

2.6 CRC

CRC calculation

All parts of the CMTP are protected by separate 8 bit block CRCs (cyclic redundancy checks). For example, the block checksum for the OTP_calibration_CRC are calculated from address range 273 to 306 using the following polynomial:

$x^8 + x^4 + x^3 + x^2 + 1$

The following CRC-seed value:

10101010B (0xAA)

When writing the second time to this CMTP area, the block checksum has to be re-calculated from the new content and written into the corresponding CRC field. In case the checksum value stored in the CRC field does not match the content, the TLE480x detects a CRC error and disables the SENT/SPC output.

Simple C implementation for the 8 bit CRC calculation:

```
/*
 This file contains a implementation of CRC8 checksum functions.
*/

#include "Crc8_J1850.h"

#ifdef CRC8_USE_TABLE
/*
 Using table from http://www.sunshine2k.de/coding/javascript/crc/crc_js.html
 Polynomial:   0x1D  (x^8 + x^4 + x^3 + x^2 + 1 polynomial of J1850)

 Initial value:   0xAA
 Final XOR value:  0xFF
*/

const uint8 crc8_table[256] =
{
 0x00,0x1D,0x3A,0x27,0x74,0x69,0x4E,0x53,0xE8,0xF5,0xD2,0xCF,0x9C,0x81,0xA6,0xBB,
 0xCD,0xD0,0xF7,0xEA,0xB9,0xA4,0x83,0x9E,0x25,0x38,0x1F,0x02,0x51,0x4C,0x6B,0x76,
 0x87,0x9A,0xBD,0xA0,0xF3,0xEE,0xC9,0xD4,0x6F,0x72,0x55,0x48,0x1B,0x06,0x21,0x3C,
 0x4A,0x57,0x70,0x6D,0x3E,0x23,0x04,0x19,0xA2,0xBF,0x98,0x85,0xD6,0xCB,0xEC,0xF1,
 0x13,0x0E,0x29,0x34,0x67,0x7A,0x5D,0x40,0xFB,0xE6,0xC1,0xDC,0x8F,0x92,0xB5,0xA8,
 0xDE,0xC3,0xE4,0xF9,0xAA,0xB7,0x90,0x8D,0x36,0x2B,0x0C,0x11,0x42,0x5F,0x78,0x65,
 0x94,0x89,0xAE,0xB3,0xE0,0xFD,0xDA,0xC7,0x7C,0x61,0x46,0x5B,0x08,0x15,0x32,0x2F,
 0x59,0x44,0x63,0x7E,0x2D,0x30,0x17,0x0A,0xB1,0xAC,0x8B,0x96,0xC5,0xD8,0xFF,0xE2,
 0x26,0x3B,0x1C,0x01,0x52,0x4F,0x68,0x75,0xCE,0xD3,0xF4,0xE9,0xBA,0xA7,0x80,0x9D,
 0xEB,0xF6,0xD1,0xCC,0x9F,0x82,0xA5,0xB8,0x03,0x1E,0x39,0x24,0x77,0x6A,0x4D,0x50,
 0xA1,0xBC,0x9B,0x86,0xD5,0xC8,0xEF,0xF2,0x49,0x54,0x73,0x6E,0x3D,0x20,0x07,0x1A,
 0x6C,0x71,0x56,0x4B,0x18,0x05,0x22,0x3F,0x84,0x99,0xBE,0xA3,0xF0,0xED,0xCA,0xD7,
 0x35,0x28,0x0F,0x12,0x41,0x5C,0x7B,0x66,0xDD,0xC0,0xE7,0xFA,0xA9,0xB4,0x93,0x8E,
 0xF8,0xE5,0xC2,0xDF,0x8C,0x91,0xB6,0xAB,0x10,0x0D,0x2A,0x37,0x64,0x79,0x5E,0x43,
 0xB2,0xAF,0x88,0x95,0xC6,0xDB,0xFC,0xE1,0x5A,0x47,0x60,0x7D,0x2E,0x33,0x14,0x09,
 0x7F,0x62,0x45,0x58,0x0B,0x16,0x31,0x2C,0x97,0x8A,0xAD,0xB0,0xE3,0xFE,0xD9,0xC4
};

uint8 Crc8(const uint8* data, unsigned int len, uint8 startvalue)
{
 unsigned int i;
 uint8 crc = startvalue;

 for (i=0; i<len; i++)
 {
  crc = crc8_table[crc ^ data[i]];
 }
 return crc;
}

#else

/* x^8 + x^4 + x^3 + x^2 + 1 polynomial of J1850 */
#define CRC8_POLYNOMIAL  (0x11D0 << 3)

uint8 Crc8(const uint8 *data, unsigned int len, uint8 startvalue)
{
  unsigned long crc = (long) ((startvalue << 8) ^ CRC8_POLYNOMIAL);
  unsigned int i, j;
  for (j = len; j; j--, data++)
  {
    crc ^= (*data << 8);
    for(i = 8; i; i--)
 {
   /* Check if the first bit is equal to 1
   In each step, the leading 1 of the polynom is aligned with the first 1 in the remainder */
     if (crc & 0x8000)
   {
  /* XOR with polynomial */
  crc ^= CRC8_POLYNOMIAL;
     }
   /* Move the first bit out */
   crc <<= 1;
    }
  }
  crc >>= 8;
  return (uint8)(crc);
}

#endif
```

Figure 27 8 bit CRC calculation Source: https://github.com/II43/Crc8_J1850/blob/master/Crc8_J1850.c

Remarks to the figure:

The CRC seed value already represents a CRC8 8-bit value. That means the data seed which run initially through the CRC8 algorithm is as follows: 01010100B (0x54)

It is also crucial to perform at the end a final XOR function on the CRC8 result with the value 0xFF.

There are the following CRC blocks:

For the ifx_crc the following block is applied to the CRC calculation:

Address range **256 to 271**.

For the otp_calibration_crc the following blocks are applied to the CRC calculation:

Address range **273 to 306** and the otp_lock bit in register 307. For the CRC calculation all other bits in 307 have to be assumed as 0.

For the otp_sec_key_crc the following block is applied to the CRC calculation:

Address range **308 to 315.**

The register 307 with the otp_calibration_crc also has an otp_lock bit. With this lock-bit it is possible to lock the otp after a write cycle if no further write cycles are needed in the future. In total 5 write cycles are possible for the otp_calibration_crc. For the registers in the range 273 to 306 only two write cycles are possible.

If the otp_lock bit is not set, but the CRC value has been written correctly the sensor will have a valid SENT/SPC output without any error indication bits. Not setting the otp_lock is only recommended in a testing environment.

3 Calibration

3.1 Angles calibration

3.1.1 Parameters for calibration

Register name in CMTP	Working register name	Parameter	Parameter range	Parameter definition
279:otp_vect_lim1_reg[max,min]	vect_lim1_reg[max,min]	N/A	0..255	**min:**0..29.5 mVp with 0.115 mV/LSB8 **max:**0..59 mVp with 0.23 mVp/LSB8
280:otp_vect_lim2_reg[max,min]	vect_lim2_reg[max,min]	N/A	0..255	**min:**0..29.5 mVp with 0.115mV/LSB8 **max:**0..59 mVp with 0.23mVp/LSB8
281: otp_offset_x1_reg	offset_x1_reg	P1	0..65535	See section 3.1.2.1
282: otp_offset_y1_reg	offset_y1_reg	P2	0..65535	
283: otp_offset_x2_reg	offset_x2_reg	P3	0..65535	
284: otp_offset_y2_reg	offset_y2_reg	P4	0..65535	
285:otp_adc_gain_reg [y2_gain,y1_gain]	adc_gain_reg [y2_gain,y1_gain]	P5 / P6	0..255	1..255LSB8 -> -0.031..+0.031, 0LSB8 =0
286: otp_adc_orto_reg [y2_orto,y1_orto]	adc_orto_reg [y2_orto,y1_orto]	P7 / P8	0..255	+/-1/64 (+/-arcsin(1/64)= +/-0.89°)
287: otp_angle_offset1_reg	angle_offset1_reg	P9	0..65535	Chapter Angle/Torque read out mapping
288: otp_angle_offset2_reg	angle_offset2_reg	P10	0..65535	
289: otp_torque_offset_reg	torque_offset_reg	P11	0..4095	Chapter Torque calibration: Torque calculation
290: otp_torque_stiffness_reg	torque_stiffness_reg	P12	0..4095	Chapter Torque calibration

3.1.2 Calibration flow

Angle / Torque calibration

The diagram below shows the flow chart of the internal datapath. The configuration parameters can be programmed via the programming interface.

Programming flow chart

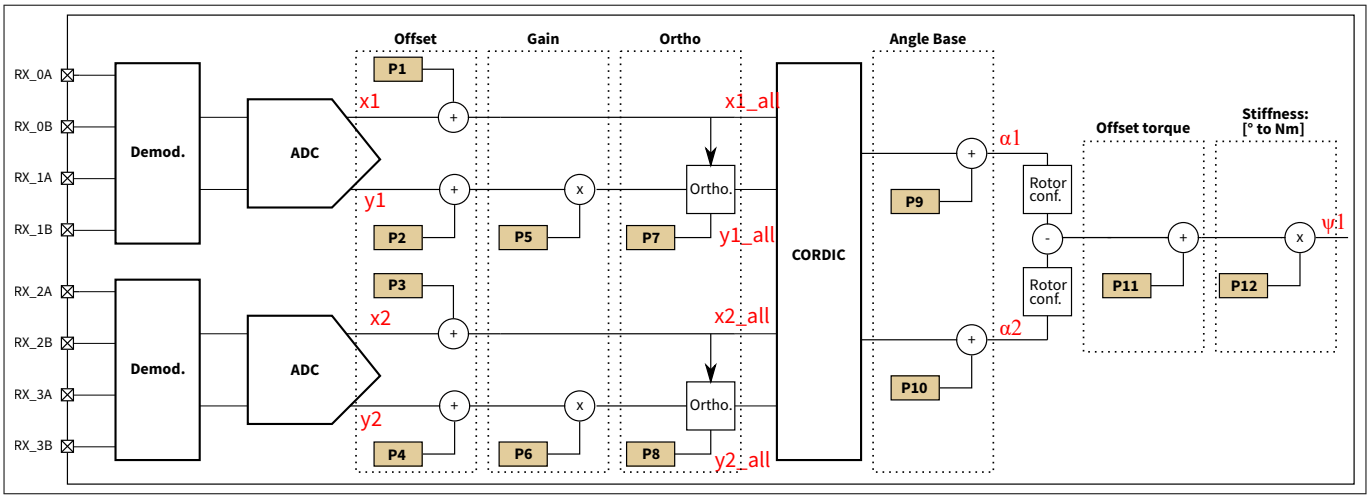

Figure 28 Programming flow chart

The coil calibration for product variants with torque output the angle 1 and angle2 coil signals are compensated.

Accuracy target of angle compensation parameters

The accuracy target of the compensation and angle calculation is an angle error of <+/-3 LSB16 in angle output.

The coil offset, gain and non orthogonality errors are compensated in the following way:

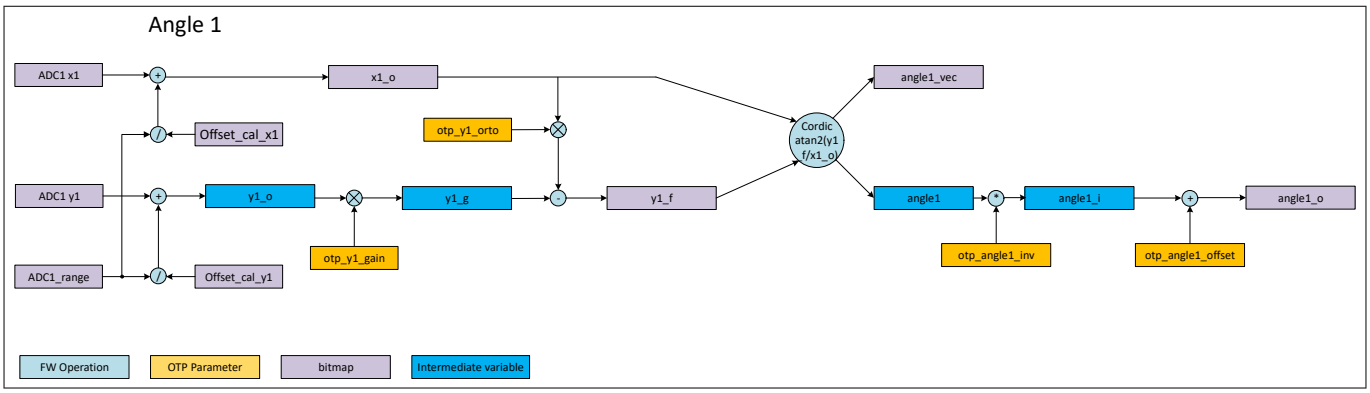

Figure 29 Angle calibration

The compensation for Angle1 and Angle2 in the main and safety path is the same. In the safety path the range is fixed to 4 as the gp_adc for the coils is always in high range.

In the following chapter the compensation is performed for the example case of angle1.

3.1.2.1 Offset compensation

The **offset compensation** consists of the following steps valid for x and y, where x1_0 and y1_o are the offset compensated values:

x1_o = x1 + (**offset_cal_x1**/adc1_range)

y1_o = y1 + (**offset_cal_y1**/adc1_range)

adc1/2_range can be 1 or 2 depending on gain range bits. The offset is always mapped to small signal ranges. (up to 28.9 mVpp signal range).

For the high range the factor is 2 and therefore only 1/2 of the offset has to be added. High range is triggered when 29028 LSB16 is reached (vector length amplitude).

3.1.2.1.1 Offset calculation

The best way to determine the electrical offset of each coil signal (x1/2, y1/2) is to compute the **integral** of the signal x and y over one electrical period or even better over a complete mechanical revolution of the target wheel.

The easiest way to compute the integral is to sum up equidistant samples (eg.: 12-24 samples for a full revolution) of x or y over exactly one revolution and divide by the number of samples.

A less accurate way is to compute the offset out of **min/max**

$$offset\left(x\right) \quad = \quad -1 \quad \times \quad \frac{max(x)+min(x)}{2} \tag{9}$$

This offset calculation is affected more by noise in min and max.

Since the ADC read and offset computation is done in LSB16 no conversion has to be applied to get the otp parameter.

The offset parameter is stored in LSBs related to the ADC small signal range. For the large ADC range or GP_ADC range the offset is divided by 2. The offset division is performed in the IC.

3.1.2.2 Gain compensation

The **gain compensation** is applied only to y, wherefore y1_g is the gain compensated value and y1_0 is the offset compensated value:

$$y1_g = y1_o \quad \times \quad \left(1+\frac{otp_y1_gain}{2^{12}}\right) \tag{10}$$

The full compensation range with 8 bit:

$$\pm\frac{1}{32} \quad = \quad \pm 0.031 \tag{11}$$

The default value for otp_y1/2_gain is 0.

3.1.2.2.1 Gain calculation

To get the **Electrical amplitude difference gain** (**otp_y1/2_gain**) compensation parameter the amplitudes of x and y coil signals have to be found. An easy and accurate way is to find the maxima and minima of each signal over a complete mechanical revolution of the system.

$$gain \quad = \frac{Ax}{Ay} \quad -1 = \quad \frac{(Max(x)-Min(x))}{(Max(y)-Min(y))}-1 \tag{12}$$

The 8 bit full scale of the OTP gain parameter covers an amplitude miss match of +/-3.1%. This means a scaling of gain by factor (32*2^7) has to be applied to get the LSB8 value.

3.1.2.3 Non orthogonality compensation

Non orthogonality compensation

The **non orthogonality** compensation is applied only to y:

$$y1_f = y1_g - x1_o \times \frac{otp_y1_orto}{2^{13}} \tag{13}$$

The full compensation range with 8 bit is +/-1/64 = +/-0.015 rad (equivalent to +/-0.89°).The default value for otp_y1/2_orto is 0.

y1_f is the sine input and x1_o is the cosine input to the cordic algorithm for the atan2 function.

3.1.2.3.1 Non orthogonality compensation calculation

After the offset and gain compensation is applied, the ADC values of x and y at 45° and 135° can be used to compensate for a not ideal 90° phase shift between x and y signal. This is called the **Electrical orthogonality compensation (otp_y1/2_ortho).**

Here it is possible to use the angle output without ortho-compensation(values which have offset and amplitude compensation applied) to position the target wheel at 45° and 135°. Deviations of the those values can be more than 10° to still allow the following algorithm a good orthogonality compensation.

M45 = vector length at 45°electrical = quadratic sum, M135 = vector length at 135° electrical = quadratic sum

$$\begin{aligned} {M_{45}}^2 &= {X_{2(45)}}^2 + {Y_{2(45)}}^2 \\ {M_{135}}^2 &= {X_{2(135)}}^2 + {Y_{2(135)}}^2 \end{aligned} \tag{14}$$

The orthogonality angle error can be calculated out of the vector length miss match:

$$\begin{aligned} ortho_error\left[°ele\right] &= 2 \times atan\left(\frac{M135 - M45}{M135 + M45}\right) \\ ortho_compensation &= sin\left(-orthoerror\right) \end{aligned} \tag{15}$$

To get the 8 bit otp parameter the calculated ortho_compensation needs to be scaled by ortho_compensation*64*2^7 to get LSB8 value. The vector length calculation on 45° and 135° has not a good performance if the angle resolution at read out is not high enough.

3.1.2.4 Vector length

The vector length extraction for angle sensors simply calculates the square root of the two signal paths (i.e. by the power of two) according to

$$l_{vec} = \sqrt{x^2 + y^2}$$

Figure 30

The actual application of a typical vector length calculation strongly depends on the receiving coils connected to the RX pins.

The system integrator is responsible for the vector length adjustment settings in accordance with the coils and the air gaps intended to use.

To further adjust the settings there are 2 registers, which can be programmed with the min and max limits of the vector length: The minimum and maximum limit separate for both vector lengths of ADC1 and ADC2.

otp_vect_lim1_reg[8bit lim_max,8bit lim_min]
otp_vect_lim2_reg[8bit lim_max,8bit lim_min]
(register 279 and register 280)

The minimum vector length limit is calculated with the minimum sensitivity of the ADC1/2 referred to coil amplitude signals in range 0:

- 1110 LSB16/mVp.

The minimum vector length covers the range 0..29.5 mVp with 0.115 mV/LSB8 resolution.

The maximum limit is derived from the ADC1/2 in range 1 with sensitivity:

- 555 LSB16/mVp

The maximum vector length covers the range 0...59 mVp with 0.23 mVp/LSB8 resolution.

The vector length of ADC1 and ADC2 has to be adjusted by the factor 2 depending on small range or medium range sensitivity.

3.1.2.5 Inverting rotation

For a general coil design also the option of inverting the rotation direction needs to be implemented:

```
if otp_angle1_inv = 1
  angle1_i = angle*-1
else
 angle1_i = angle
```

To the angle1 output of the cordic an **otp_angle1_offset** is added to compensate for mechanical rotor angle mounting tolerances.

- angle1_o = angle1 + **otp_angle1_offset**

3.1.3 Torque calibration

Torque calibration

For the torque calibration as first step it is possible to set an offset value for the differential angle calculated out of angle1 and angle2.
To calculate a torque value a multiplication with the torque stiffness value is necessary. This way the torque output can be calibrated according to customer chosen torque bar and mounting conditions.

Parameter:

- **otp_wings1/2** = 5 bit -> clamped between 1.. 24
- **otp_torque_offset** = 12 bit to cover +/-25 % of the diff_angle range.
- **otp_torque_stiffness** = 12 bit to trim to a desired LSB/Nm torque sensitivity from 1.4 .. 0.4.

The diff_angle is calculated as follows:

$$Diff_angle = Mod\ \left(Angle1_mc \times otp_wings2 - Angle2_mc \times otp_wings1; 2^{16}{}_{(360°)}\right) \quad (16)$$

Angle1_mc and Angle2_mc are 16 bit values for 0 .. 360° electrical angle out of the angle calibration.
The diff_angle result is a 16 bit value for a differential mechanical angle range of $2^{16}{}_{(360°)}$/ (opt_wing1*otp_wings2).

The TLE4802 firmware does not consider the common divider. The number of wings do not necessarily correspond to the mechanical wings:
In the case of a common divider in the number of wings the uniqueness range is reduced by the factor of the common divider.
To obtain the full uniqueness range use the the factor of the wings/common divider.

Example:

wing1 = 4, wing2 =10 -> programming wing1 = 2 and wing2 = 5 is possible. The implemented algorithm will not remove the common divider. And without a common divider the uniqueness range can be adapted to the use case.

Torque calculation

The calculation to get the 12 bit value to be transmitted over SENT/SPC consists of an offset summation to the diff_angle and the stiffness factor multiplication. The otp_torque_offset and otp_torque_stiffness are calibrated at customer side.

$$torque_o_comp = diff_angle + otp_torque_offset \times 8 \quad (17)$$

$$torque = torque_o_comp \times \frac{(3686 + otp_torque_stiffness)}{2^{12}} \quad (18)$$

The stiffness multiplication factor is in the range of 1.4 ..0.4 --> 0.9 +/- 0.5.

If due to the multiplication factor an overflow occurs, the value will be clamped to the highest LSB12 value. Then the error code 4094 will be sent out in the SENT/SPC data frame for the torque information.

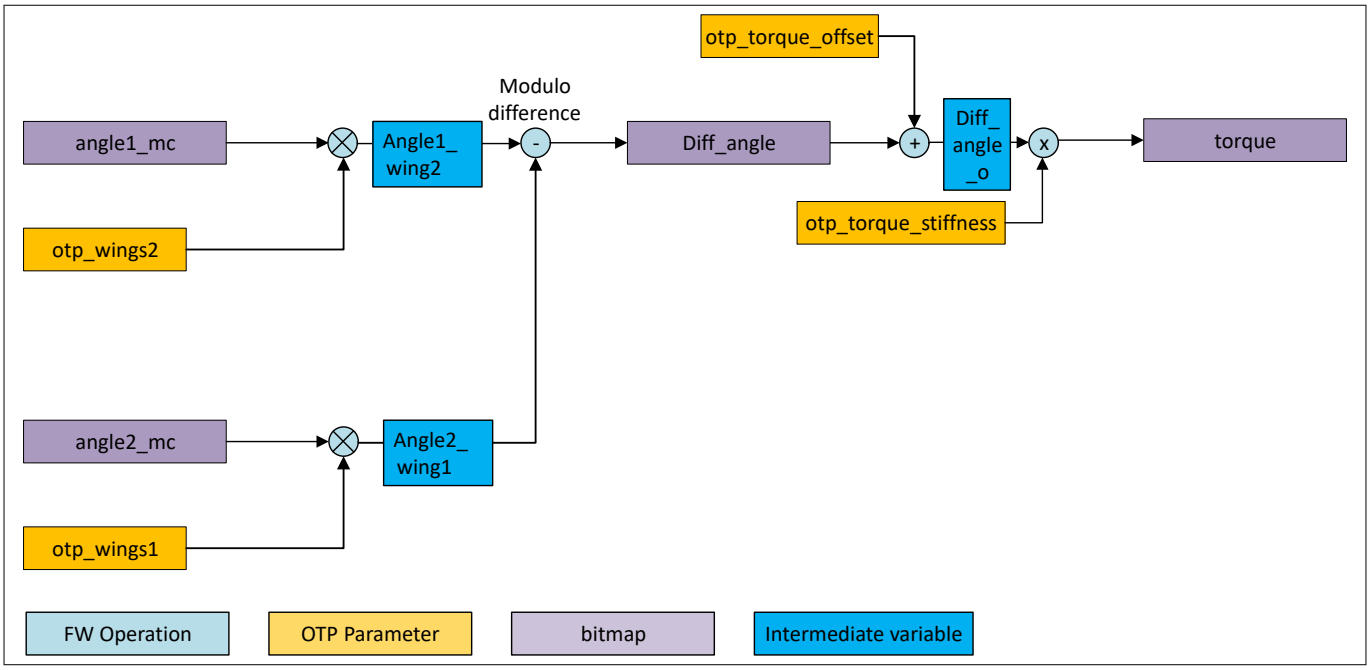

Figure 31 Torque calibration

Remarks to the figure:

Torque is referred here as a difference angle and as torque if stiffness is not 1.

Mapping for different SENT/SPC frame formats:

The calculation to map the full data range to the 12 bit range can handled with the below functions. First step beforehand is to truncate the original 16 bit data to 12 bit MSB-aligned data.
Afterwards the mapping can be performed as follows:

The calculation to map the full data range to the 12 bit range is handled with

$$sent_data = data - \frac{data + 1}{2^9} + 1 \tag{19}$$

3.1.3.1 Difference angle uniqueness

The uniqueness of the difference angle is to be calculated based on the otp_wings1 and otp_wings2.
The difference angle is calculated with the known formula from the previous chapter:

$$Diff_angle = Mod\ \left(Angle1_mc \times otp_wings2 - Angle2_mc \times otp_wings1; 2^{16}{}_{(360°)}\right) \tag{20}$$

Angle1_mc and Angle2_mc represent in this equation the electrical angles.
The maximum range can be calculated by the following formula:

$$Diff_angle_range = \frac{360°}{wings1 \times wings2} \times GCD\left(wing1; wing2\right) \quad (21)$$

GCD: Greatest common divisor.

By using the GCD, it is possible to increase the uniqueness range. To enable in the IC this maximum range, the described principal from above needs to be applied.

Examples:

otp_wings1 = 2, mechanical wings on target/sine/cosine periods on PCB = 6
otp_wings2 = 3, mechanical wings on target/sine/cosine periods on PCB = 9

Uniqueness:

$$\frac{360°}{6 \times 9} \times 3 = \pm 10° \quad (22)$$

If the same number of wings would have been applied in the above example we get the following result:

otp_wings1 = periods1 = 6
otp_wings2 = periods2 = 9

Uniqueness:

$$\frac{360°}{6 \times 9} = \pm 3.33° \quad (23)$$

3.2 Lookup table

Lookup table usage

The device embeds a linearization block to compensate for non-linearity behavior (e.g. coils/rotor misalignment). Two options are possible:

- An 8-point look-up table using a 2D implementation where the x and y values can be written independently. This implementation allows the use of non-equidistant points to optimize the linearization process only where it is required.
- Two 8-point look-up tables using a 1D implementation where the x values are equidistant.

With both options, between the pivot points Pn (nx,yn), a piece-wise linear interpolation is performed.

If no linearization is required, the following options are available:

- First option would be to program the linear slope with the supporting points on the slope - this only applies for the 1D LUT: dual angle output.
- Second option would be to program the first and last value with a "0". Writing "0" values to the corresponding registers: 1D and 2D look up table.

The firmware only checks first and last value when checking to apply the LUT.

Lookup table limitations

The lookup table steepness is limited to 2.

- The output can therefore change no more than 2 times faster than the input.
- The equivalent mapping of the lookup table can therefore only be done from 180° to 360° and for example 90 ° to 360° is not possible.

3.2.1 1D lookup table

3.2.1.1 1D lookup tables for dual linear angle output

1D lookup tables for dual linear angle output

The 1D look up table variant is used for dual angle output. For the torque calculation the following compensation is not applied.

The otp registers for the LUT are 16 bit each, so the following assignment is assumed for angle1 and angle2:

otp_lut_0..7 = 8*16 bit LUT for angle1
otp_lut_8..15 = 8*16 bit LUT for angle2

The 1D lookup table with 8 points is applied only in the case of dual angle application use case with no torque calculation.

The linearization applied is a piece wise linearization, which can be described as follows:

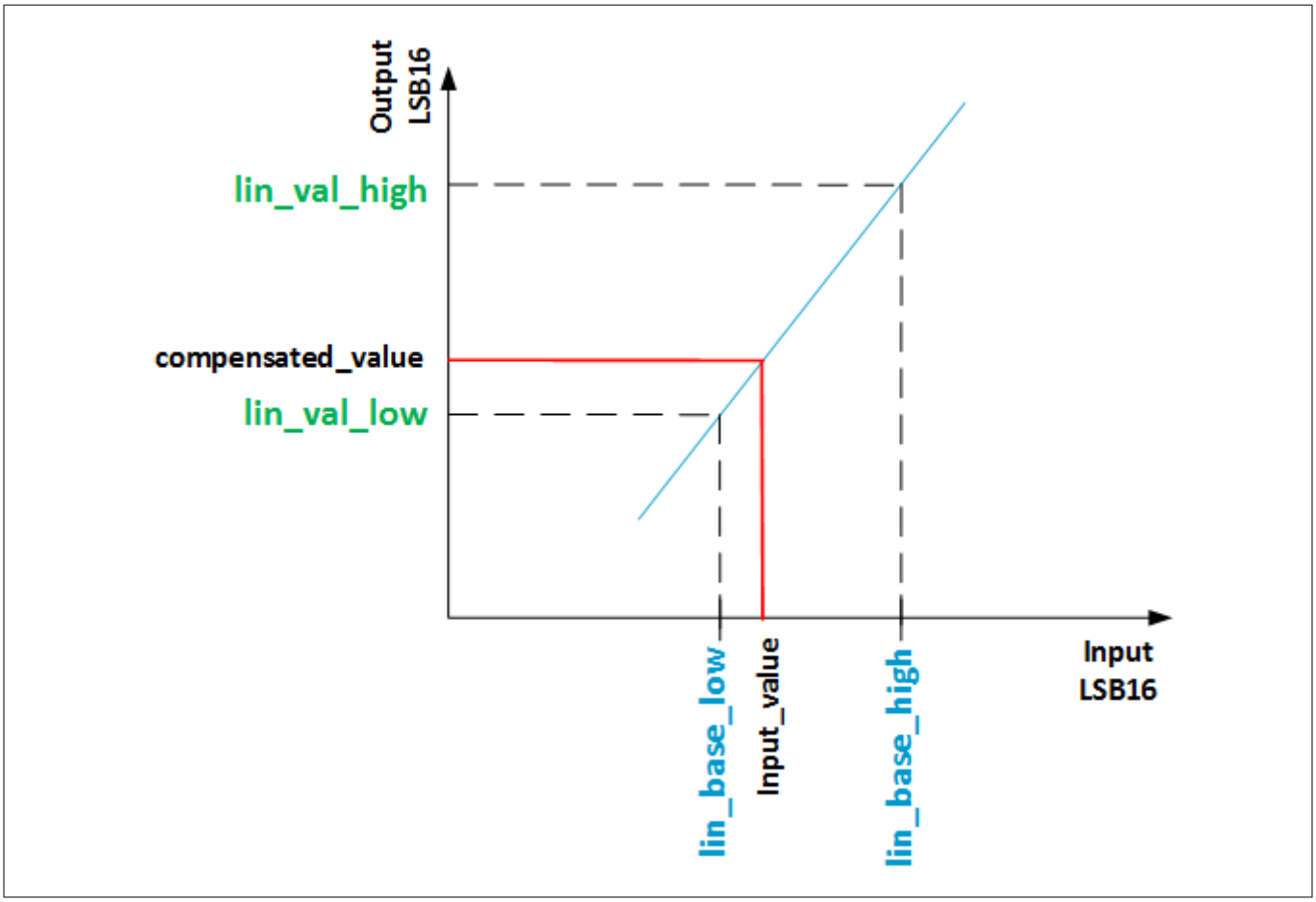

Figure 32 General "linearization element" for piece wise linearization.

The input_value are the offset compensated angle values (angle1_o, angle2_o), which are within one pair, i.e. lin_base_low and lin_base_high of the 8 points defining a "linearization element". With respect to the two programmed values lin_val_low and lin_val_high, a slope can be computed which represents the compensation slope for this range. Mathematically, this can be represented as:

- compensated_value = (input - n*8192)*(lin_val_high-lin_val_low)/8192 + lin_val_low

The compensated values are written into the the angle1_mc and angle2_mc register.

The next figure shows the arrangement of the points for equally spaced linearization points for the two angles

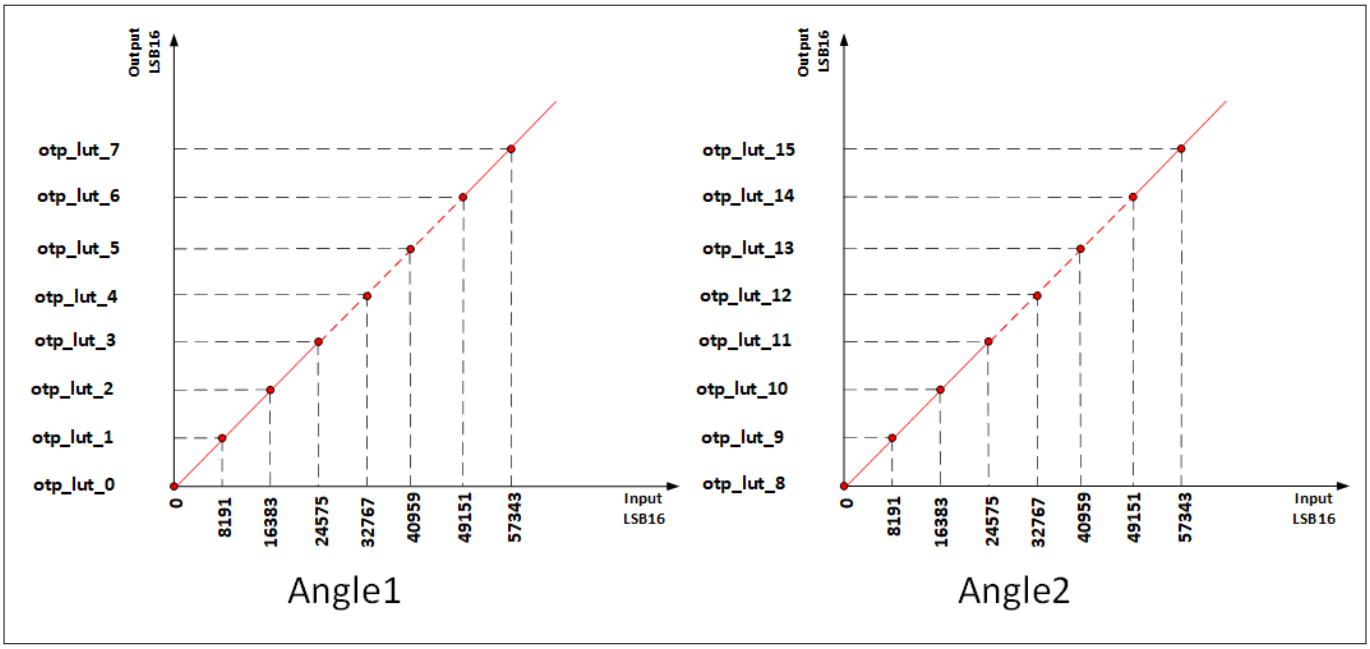

Figure 33 Linearization points for Angle 1 and Angle 2

When no linearization is needed there are two options: First option would be to program the linear slope as depicted above with the supporting points on the slope - this only applies for the 1d LUT: dual angle output. Second option would be to write "0" values to the corresponding registers.

3.2.1.2 Mechanical angle error compensation with 1D look up tables

Mechanical angle error compensation with 1D look up tables

The mechanical angle error compensation is applied with torque output. The compensation is executed on the angles calculated in the safety path and the standard signal path.

Parameter:

For the compensation the following otp registers are relevant:

otp_wings1/2 = 5 bit
otp_lut_0..7 = 8*16 bit
otp_lut_8..15 = 8*16 bit
otp_mech_inv_dir = 1 bit

Limitation for the compensation:
the maximum number of wings on rotor 1 is 24.
The typical nonius calculation is done with a difference of **otp_wings1** - **otp_wings2** = +/-1.
The nonius calculation over full mechanical angle will not work if **otp_wings1** and **otp_wings2** have a common divider.
To achieve a mechanical angle output in torque channel the otp_wings1/2 need to be programmed as **1** each.

To be able to compensate the first harmonic of mechanical angle errors the mechanical angle has to be calculated inside the IC.

3 Calibration

The mechanical angle consists of the wing number plus the electrical angle inside the wing. As reference rotor the rotor 1 (angle1) is used.
360° in general is represented as 2^16 since all operation are referred to to 16 bit operations.

Wing1/2_pos:

The wing1_pos can be calculated in 16 bit operations if the Angle1_o and Angle2_o are shifted by 4 positions (15 ..4).
All combination with difference +1 or -1 are working. A wing pair like 2 and 9 would work. But the vice verse would not work. Desired wing pairing should be verified when checking mechanical angle output.

Mechanical angle:

The mechanical angle is the sum/difference of the anlge1_o and the wing1 position. To be able to stay in 16 bit operations the 4 LSB can be truncated.

if **otp_mech_inv_dir = 0**
 mech_angle = (Angle1_o + $2^{16}_{(360°)}$*wing1_pos)/**otp_wings1**
else
 mech_angle = (-Angle1_o + $2^{16}_{(360°)}$*wing1_pos)/**otp_wings1**

Mechanical angle error lookup table

The angle error lookup table based on the mechanical angle is an equidistant 8 point lookup table starting from 0, 1/8 .. to 7/8 of the mechanical angle 360°. The error stored in the 1d lookup tables is an electrical angle for angle1 and angle2 based on the mechanical position.
Two errors have to be calculated out of the two lookup tables after the position n within the table is found. The position n is found by hardware. Input value is the mech_angle calculated from angle1 and angle2.
lin_base_low and lin_base_high are the segment borders n*$2^{16}_{(360°)}$/8 and (n+1)$2^{16}_{(360°)}$/8 so that mech_angle falls between.

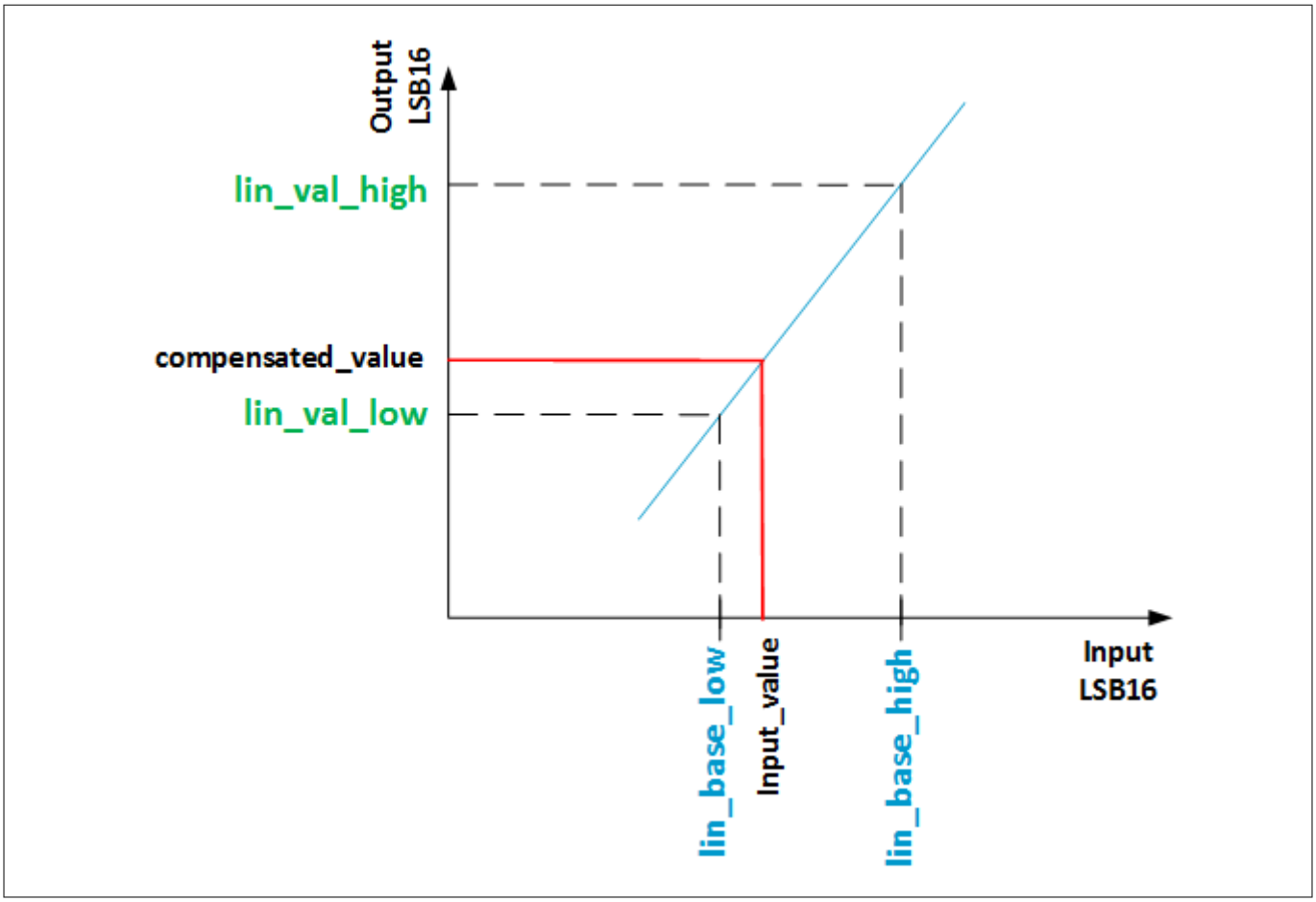

Figure 34 LUT input - output graph

Figure: General "linearization element" for piece wise linearization.

error1_mech = (input - n*8192)*(lin_val_high1-lin_val_low1)/8192 + lin_val_low1
error2_mech = (input - n*8192)*(lin_val_high2-lin_val_low2)/8192 + lin_val_low2

lin_val_high1 and lin_val_low1 are values out of the **otp_lut_0..7**
lin_val_high2 and lin_val_low2 are values out of the **otp_lut_8..15**

Error compensation:

angle1_mc = angle1_o - error1_mech
angle2_mc = angle2_o -error2_mech

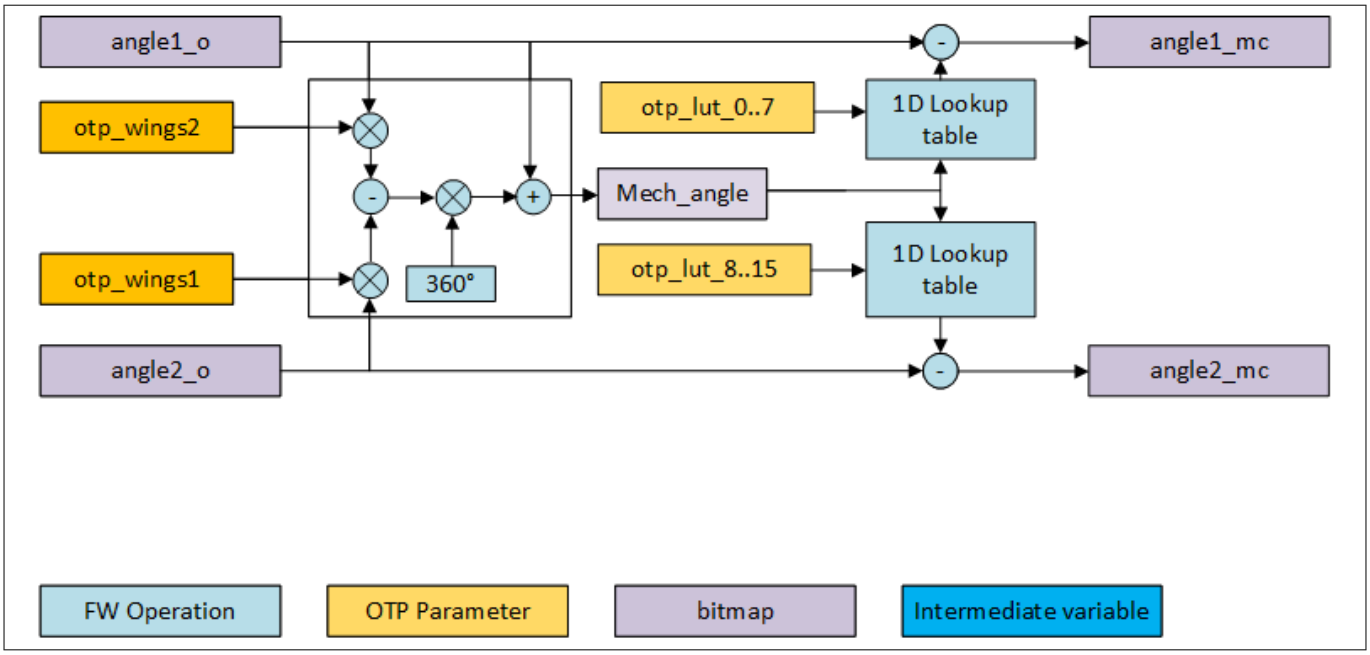

Figure 35 Sequence diagram for the mechanical angle compensation.

Important remarks about the usage of the Look up table when as first and last value zero is applied: the firmware does not apply the mapping of the look up table for mechanical angle error compensation when the first and last value is zero. The internal check only checks for the first and last value. One of them needs to be greater than 0 to apply the compensation for the mechanical angle calculation.

4 Timing model

4.1 Timing SENT/SPC

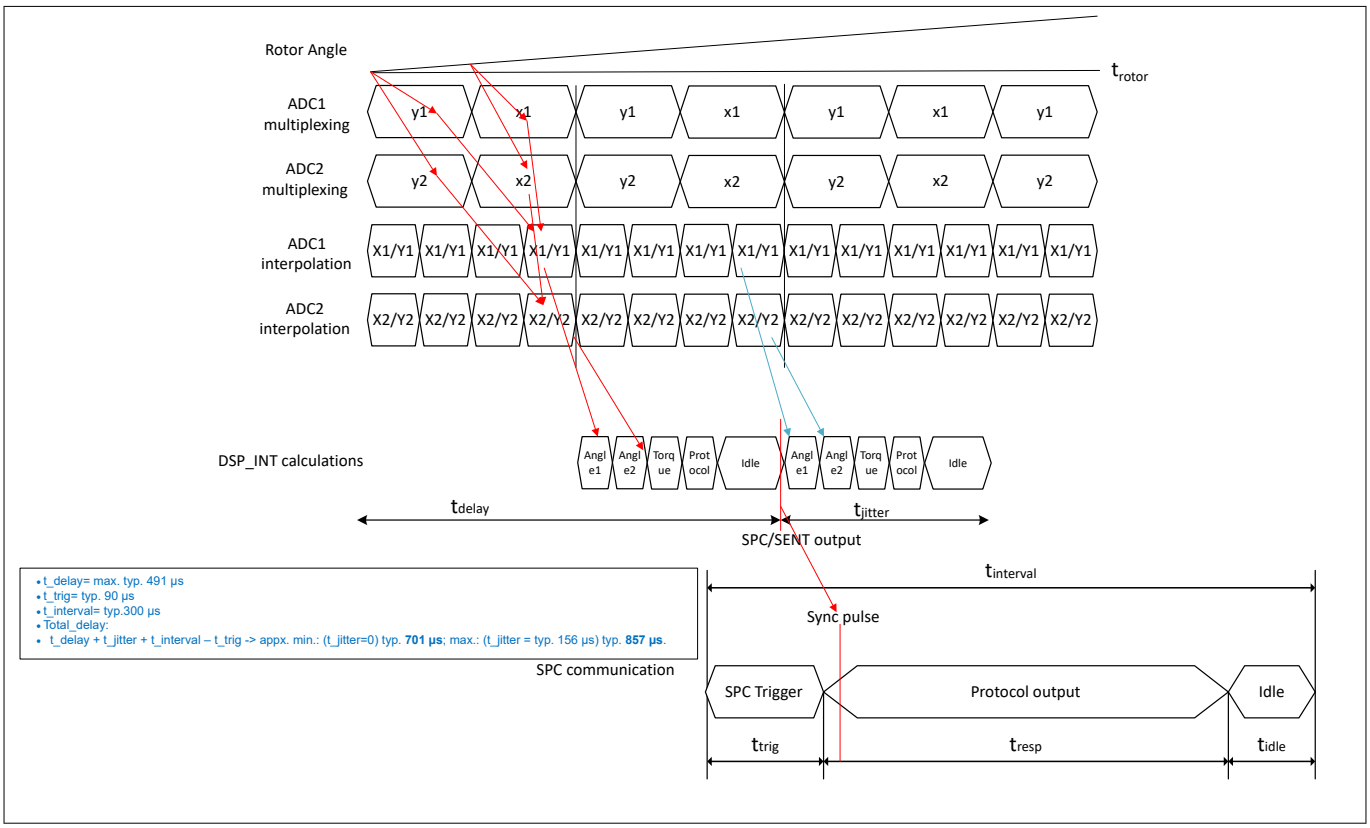

Figure 36 SENT/SPC Interface timing diagram

The signal acquisition of ADC1 and ADC2 is in parallel. The multiplexing between the input channels x1/x2 and y1/y2 is interpolated in parallel with a remaining maximum delay of typical 2µs.

The angles/torque values calculated in the DSP_INT have an update rate of typical 156µs.

There are two modes when looking into the timing diagram of the TLE480x:

Synchronous mode:

In synchronous mode (Pause pulse activated) the SENT transmission is synchronized with t_delay and the pause pulse.

For UT > 1 µs pause pulse is adjusted to synchronize with t_delay. This mode is only applicable to SENT output.

Asynchronous mode:

In asynchronous mode the t_jitter need to be added as described also under Timings_SPC for total delay calculations.

Asynchronous SENT and SPC have the same delay/jitter.

Internal sampling time is always constant and can not be changed. Tick time configuration only affects the SENT frame length.

4.2 SICI delay timing model for calibration procedures

SICI delay timing model for calibration procedures

Timing between angles:

- The latched X1/Y1 value has a filter delay of 280 µs additionally. This value is constant.
- The 156 µs jitter is a maximum value and could be also much less. A fit over multiple points is necessary to enable a small delay when calibrating to a reference angle sensor.

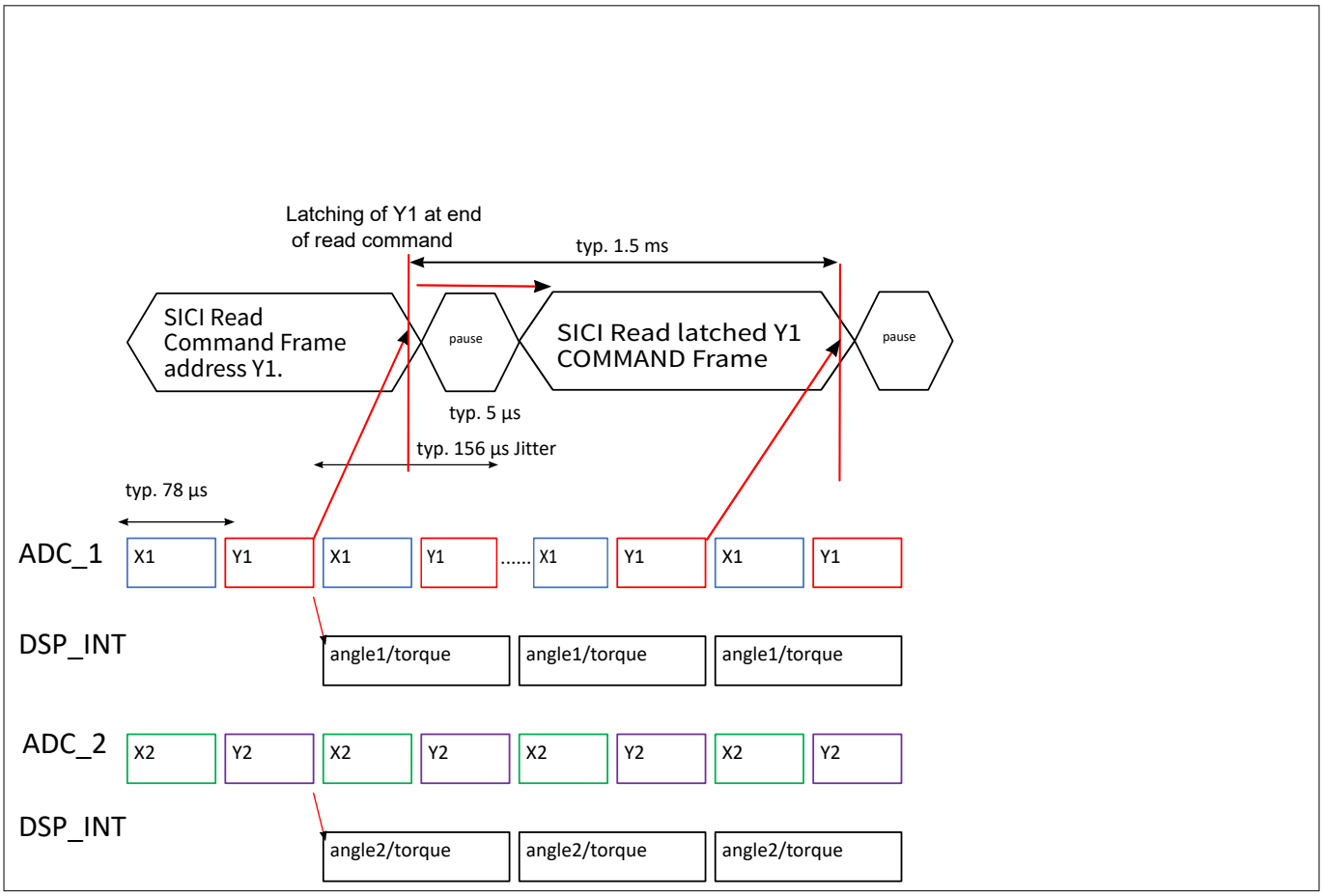

Figure 37 SICI Read delay model

5 Parameter mapping

5.1 Short serial message items

5.1.1 Message IDs

Message ID mapping - each message ID has 16 bits

Message-ID 0	otp_chip_ID_3 of bit map
Message-ID 1	otp_chip_ID_2 of bit map
Message-ID 2	otp_chip_ID_1 of bit map
Message ID 3	The 8 bit temperature output Temp[7:0] is calculated as tlin1/64. This gives an resolution of 2°C/LSB8. The 8 MSBs in the Message ID 3 are 0.
Message ID 4	otp_ssm_custom1
Message ID 5	otp_ssm_custom2
Message ID 6	otp_ssm_custom3
Message ID 7	The 8 bit vector length from RX_2/3 (ADC2) and 8 bit from RX_1/2 (ADC1) covers the range 50 mVp with a resolution of 0.461 mV/LSB8.
Message ID 8 -15	values of ID0-ID7 are repeated.

5.1.2 Temperature mapping

Temperature mapping

The temperature value in the short serial message has the following LSB8 range and resolution:
2°C/LSB8 -> -256°C bis +256° , 0°C is at zero LSB (0LSB8) and in the range 1LSB8- 127LSB8 we have the positive temperature values, from 128LSB8-255LSB8 we have the negative temperature values.

5.2 Angle/Torque read out mapping

Angle/Torque read out mapping

Angle/Torque(diff-angle when stiffness =1) value read-out with SICI, register 4-5 and register 8(angle1_dsp_int_reg,angle2_dsp_int_reg,torque_dsp_int_reg)

Please refer to Command and response frame on how to read registers with SICI. In the register 4, the angle value is stored as a 16-Bit value in bit [15:0]. It is described as a signed register, the angle value is stored as Two's complement.

A Two's complement number is generated by the following formula:

$$Value = -b_{MSB} \times 2^{N-1} + \sum_{i=0}^{N-2} b_i \times 2^i \quad (24)$$

Example:
For a register value of $(1000\ 1101\ 1001\ 0011)_B$, the Two´s complement is calculated as:
$(-(1000\ 0000\ 0000\ 0000)_B + (1101\ 1001\ 0011)_B = -32768 + 3475 = -29293$

The corresponding angle calculates to:

$$Angle\left[°\right] = \frac{360°}{2^{16}} \times ANG_VAL\left[digits\right] = \frac{360°}{65536} \times \left(-29293\right) = -160.91° \quad (25)$$

It is also possible to calculate the angle in an alternative way not using Two's complement. With the 16-Bit resolution, 1 LSB corresponds to $360°/2^{16} = 360°/65536$. A register value of $(1000\ 1101\ 1001\ 0011)_B$ equals to 36243 LSB_{16}, thus giving an angle of:

$$Angle\left[°\right] = \frac{360°}{65536} \times readout\ \left[LSB_{16}\right] = \frac{36243 \times 360°}{65536} = 199.09° \quad (26)$$

This results in the same angle as 199.09°= -160.01°.

Angle value read-out with SENT/SPC

With the SENT/SPC interface, the angle value can be selected as a 16, 12 or 10-bit value.
In each case the correspondent angle value has to be calculated according to the selected output resolution.
For example for 12-bit value (1 ... 4088).
1 LSB equals an angle of $360° * 1/2^{12} = 0.0879°$ A read-out of "1" corresponds to an angle of 0°, an read-out of "4088" equals an angle of 360°.

6 Abbreviations

Abbreviation	Definition	
CMTP	Customer Multiple Time Programmable	
ECU	Electric steering Control Unit	
PWM	Pulse Wide Modulation	
SENT	Single Edge Nibble Transmission - protocol	
SSM	Short Serial Message	
SPC	Short PWM CODE	
UT	Unit time	

7 References

The following documents provide more information and referenced parameters:

[1] Latest Infineon-TLE4801-TDS datasheet

8 Revision History

Revision	Date	Change description
1.0	2024-07-17	Product release.
1.01	2024-12-02	B21 product release.

Trademarks

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

Edition 2024-12-02

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2024 Infineon Technologies AG

All Rights Reserved.

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

Email: erratum@infineon.com

Document reference
IFX-byt1732201579641

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

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

Warnings

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

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