TMF8806 host driver communication – I²C commands

Application Note

Published by ams-OSRAM AG
Tobelbader Strasse 30,
8141 Premstaetten Austria
Phone +43 3136 500-0
ams-osram.com

TMF8806 host driver communication – I²C commands

Application Note No. AN001069

Valid for:
TMF8806

Abstract

This document specifies the I²C communication between TMF8806 and an I²C host.

It describes the sensor start sequence, running measurements, calibration and retrieving diagnostic information (e.g. histograms).

In addition, it shows procedures to increase measurement accuracy with oscillator trimming and oscillator drift correction.

Information for low power operation and running several sensors on a single I²C bus is also provided.

Table of contents

1 Introduction

This document describes the I²C communication between TMF8806 and an I²C host.

Please refer to the datasheet for the voltage level that corresponds to a logic HIGH and a logic LOW. Throughout this document the term high/low is used to describe the corresponding voltage levels. No numbers are given.

The primary interface is I²C. The default I²C 7-bit slave address (unshifted) is 0x41. For description of I²C sequences the following nomenclature will be used in this document:

S Start condition

W Write direction

R Read direction

P Stop condition

Sr Repeated Start condition

A Acknowledge transmitted

N Not acknowledge transmitted

The sequences given are all for the host to be transmitted. All data is given in hexadecimal format without leading 0x.

Example to write to register 0x08 the value 0x12:

S 41 W 08 12 P

Example to read 4 bytes from register 08:

S 41 W 08 Sr 41 R A A A N P

If a single I²C master is used the last could also be safely written as:

S 41 W 08 P S 41 R A A A N P

If you are using a multi-master bus the above should not be used, as a different master may have reset the register select address.

2 General flow

Here's the minimum number of required steps to start a measurement with the TMF8806 and to read a result:

1. Power up the TMF8806 (VDD)
2. Pull the enable line high
3. Wait for at least 1.6ms (bootloader startup time)
4. Poll register 0xE0 until the value 0x00 is read back (wait for bootloader sleep)
5. Write the value 0x1 to register 0xE0 (ENABLE) (see chapter 4)

   S 41 W E0 01 P
6. Poll register 0xE0 until the value 0x41 is read back (see chapter 5.1)

   S 41 W E0 Sr 41 R A P
7. Write the value 0xC0 to register 0x02 to request the ROM application firmware (App0) start

   S 41 W 02 C0 P
8. Poll register 0x00 until the value 0xC0 is read back

S 41 W 00 Sr 41 R A P

9. Start distance measurements with default settings

10. S 41 W 06 00 00 11 02 00 00 06 1E 84 03 02 P

11. Poll register 0xE1 until the value 0x01 is read back (see chapter 8.3)

    S41 W E1 Sr 41 R A P

12. Write the value 0x01 to register 0xE1 to clear the result interrupt

13. Read registers 0x1D to 0x23 to get the measurement result (see datasheet for more details)

Table 1: Measurement result fields

Register address	Register name	Description
0x1D	STATUS	Value < 0x10 measurement OK, > 0x0F measurement error
0x1E	REGISTER_CONTENTS	Value == 0x55 for measurement results
0x1F	TID	Unique transaction ID
0x20	RESULT_NUMBER	Result number
0x21	RESULT_INFO	Object detection reliability and measurement status
0x22	DISTANCE_PEAK_0	Object distance LSB
0x23	DISTANCE_PEAK_1	Object distance MSB

The chapters below give more details about the different steps and the I²C strings to be sent/ received.

3 Startup

This chapter describes the startup behavior of the TMF8806 and also how the host should communicate with the TMF8806 during and immediately after the startup.

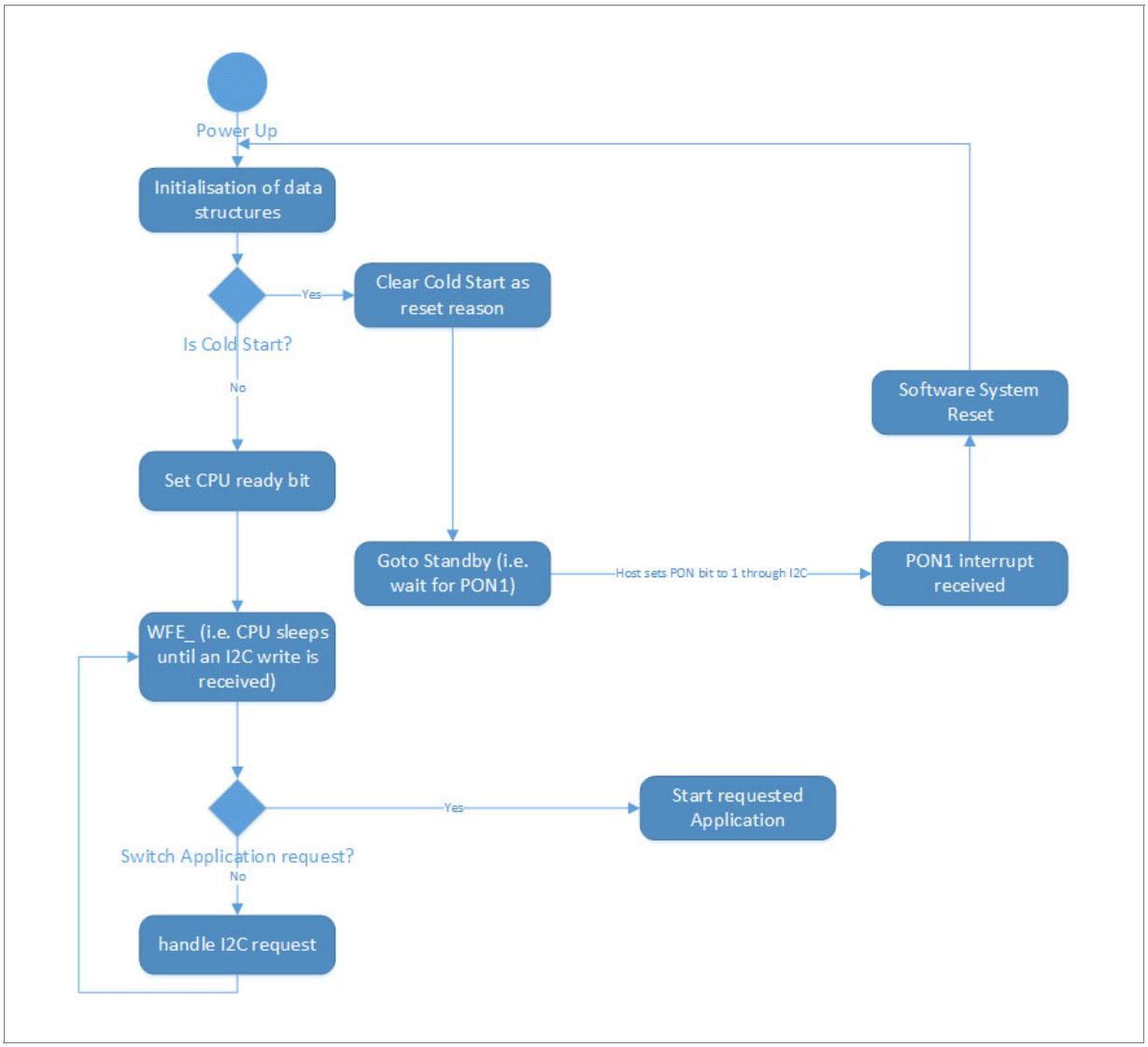
3.1 TMF8806 startup behavior

TMF8806 has 2 different applications built into ROM.

- Bootloader application (I²C Register 0x00 has the value 0x80)
- App0 application (I²C Register 0x00 has the value 0xC0)

After power-up the bootloader application is started automatically. The decisions the bootloader makes at startup are shown in the figure 1 below.

Figure 1: Bootloader startup behavior



The bootloader application keeps running and waits for inputs from a host via I²C.

3.2 Host behavior at and after TMF8806 startup

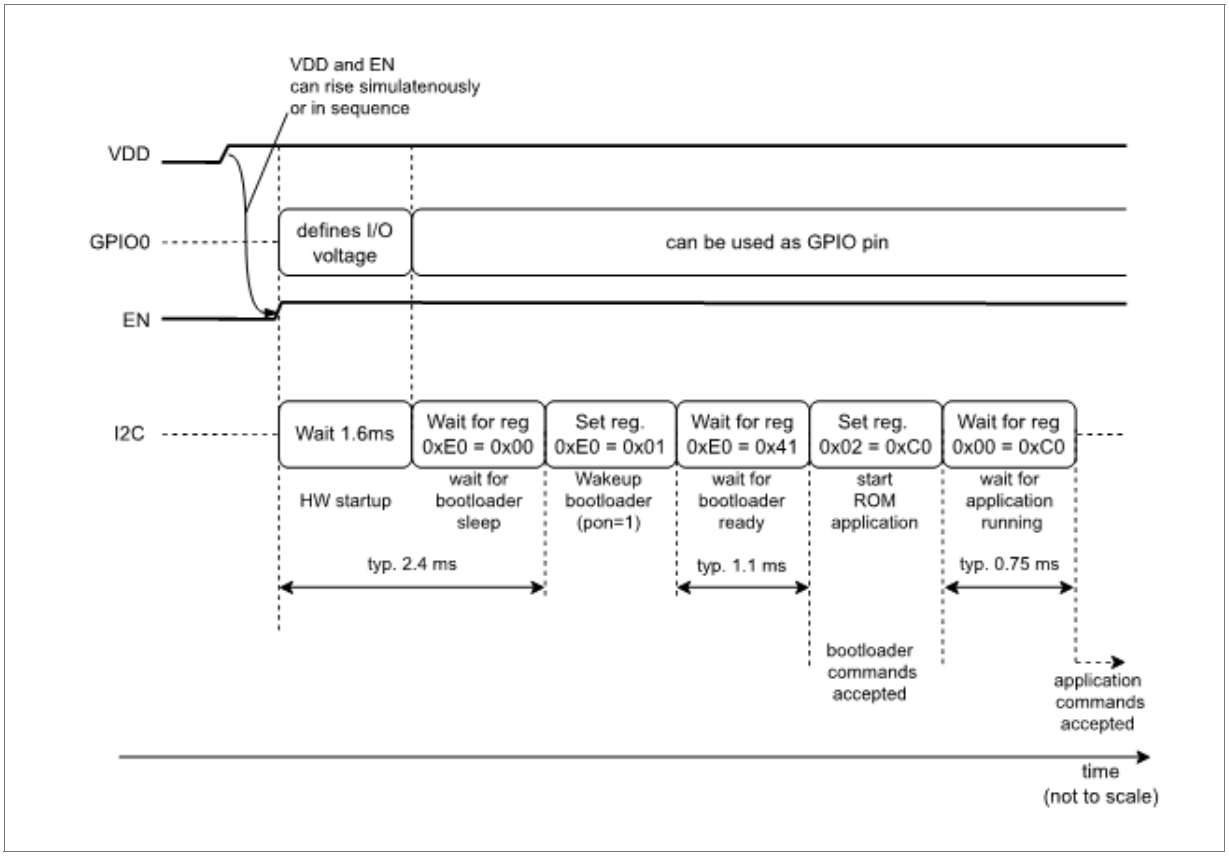
The host can control the TMF8806 through the following lines: power (VDD), ENABLE, SDA, SCL, GPIO0 and GPIO1.

The device can inform the host through the following line: Interrupt (INT).

If the host powers the TMF8806 and sets the ENABLE line high, the TMF8806 will perform the startup sequence according to the description given in chapter 3.1.

Afterwards the host should follow the control flow given in the picture below:

Figure 2: Host behavior after ToF power up/setting enable line high



Information:

The above figure is only true if the TMF8806 has just been powered up or the ENABLE line has been just set high (see TMF8806 datasheet for further details).

4 TMF8806 state – at any time

To find out the state of the TMF8806 or to put the TMF8806 into a specific state the following I²C sequences should be used.

Attention:

I²C SW registers (0x00-0xDF) shall only be read or written when the CPU is ready. Otherwise the behavior of the TMF8806 is undefined, and the read results are unreliable.

4.1 Is the TMF8806 CPU ready?

To find out if the CPU is ready use the following sequence:

S 41 W E0 Sr 41 R A P

- If the read value is 0x41 the CPU is ready and running.
- If the read value is 0x01 the CPU is running but the application is not ready to receive I²C requests. I.e. the application has not yet set the cpu_ready bit.
- If the read value is 0x00 the TMF8806 is in standby state and can only be woken up by the sequence given in section 4.2.
- If you read any other value check the datasheet.

Information

You may have to read the cpu ready bit several times if you are receiving 0x01, after startup if you are using a quick host.

Information

The host should only talk to the I²C SW registers 0x00 – 0xDF when the CPU ready bit is set and the TMF8806 is not in standby state. All TMF8806 applications clear the CPU ready bit before entering standby state. Please check the datasheet for details.

4.2 Wake up from standby state (PON=1)

To wake up the TMF8806 after it has entered the standby state you have to send the following I²C sequence from the host:

S 41 W E0 01 P

This will trigger the PON1 interrupt in the TMF8806 and the interrupt service routine (ISR) for PON1 will perform a software reset. After that the application should be ready within a few milliseconds.

4.3 Put the device into standby state (PON=0)

To put the device into standby state you have to send the following I²C sequence from the host:

S 41 W E0 00 P

This will trigger the PON0 interrupt in the ToF device and the ISR for PON0 will call the standby function. This function shuts down the sensor hardware blocks and enables the PON1 interrupt.

5 Talk to TMF8806

Once the CPU is ready all I²C registers are accessible for the host. There is no need to continuously check the content of the 0xE0 register as the host controls the behavior of the TMF8806. The TMF8806 will only change the content of the 0xE0 register when requested to do so by the host.

5.1 Find out which application is running

The I²C register 0x00 gives information about the currently running application. Use the following I²C sequence:

S 41 W 00 Sr 41 R N P

If the read back value is 0x80 the bootloader is running, if it is 0xC0 the application firmware (App0) is running.

Reading the content of register 0x01 gives the major version of the currently executed application.

S 41 W 01 Sr 41 R N P

The TMF8806 ROM Bootloader has version 0x11.

6 Bootloader protocol

The bootloader/monitor supports a binary protocol to:

- Download a program to the internal RAM.
- Perform ROM/RAM remap and reset to start the application firmware downloaded to RAM.
- Switch to the application firmware (App0) in ROM (please see chapter 2).

As the TMF8806 can be operated directly from the on-chip ROM, only minimal control of the bootloader is required. To run the ROM application, follow this procedure:

- Check that the TMF8806 CPU is ready (see chapter 4.1) and the bootloader is running (see chapter 5.1).
- Write 0xC0 to the APPREQID register 0x02
  S 41 W 02 C0 P
- Poll register APPID until you read the value 0xC0.
- Please also refer to the datasheet chapter “Startup of TMF8806”.

6.1 Command overview

The bootloader protocol is implemented through an I²C register map. The following commands are supported by the bootloader:

Table 2: Command overview

Command	Value	Meaning
Reserved	0x0 .. 0xF	Reserved, used for status reporting
Reset	0x10	Perform software reset
RAMREMAP_RESET	0x11	Remap RAM to address 0 and reset
ROMREMAP_RESET	0x12	Remap ROM to 0 and start bootloader
W_RAM	0x41	Write RAM region (plain = not encoded into e.g. Intel Hex Records)
ADDR_RAM	0x43	Set the RAM write pointer to a given address
Reserved	All other values	Ignored by bootloader / not to be used

6.2 Status

The allowed value range for status information is 0x0 ... 0xF. Therefore, any status value indicating the completion of a command (successful or unsuccessful) can be distinguished from a pending command. (Command range is 0x10 ... 0xFF with some values that are reserved). See also section 6.1 for further information about command values.

The following status values are supported by the bootloader:

Table 3: Status overview

Command	Value	Meaning
READY	0x0	Bootloader is ready to receive a new command
ERR_SIZE	0x1	The size field has an invalid number (e.g. Reset command must have size set to 0, any other value will lead to this error) and bootloader is ready to receive a new command.
ERR_CSUM	0x2	The checksum is wrong and the bootloader is ready to receive a new command.
ERR_RES	0x3	The command is not supported by the bootloader and the bootloader is ready to receive a new command.
ERR_APP	0x4	Application switch not supported and the bootloader is ready to receive a new command.
ERR_TIMEOUT	0x5	Timeout occurred and the bootloader is ready to receive a new command.
ERR_LOCK	0x6	The command cannot be executed on an encrypted device and the bootloader is ready to receive a new command (deprecated).
ERR_RANGE	0x7	The specified address is out of range or the command would lead to a read/write to an out-of-bounds address and the bootloader is ready to receive a new command.
ERR_MORE	0x8	The command was executed but did not lead to a success. For each command that can return this error code, the section specifies how to interpret the additional information. The bootloader is ready to receive a new command.
ERROR	0x9..0xF	Unspecified error and the bootloader is ready to receive a new command
BUSY	0x10 .. 0xFF	Bootloader cannot receive a new command – wait for status to become READY or ERROR

For all commands: if the command is incorrect, the command is not executed and an error code (see Table 3) is flagged in the CMD_STAT register.

If the command is valid, it is executed by the bootloader and the READY return code is flagged in the CMD_STAT register.

If either READY or an error code is flagged in the CMD_STAT register, the bootloader is ready to receive a new command.

The host must wait until BUSY (values 0x10 … 0xFF) is changed to READY or ERR_* before it may issue a new command.

6.3 Checksum calculation

Take the sum of CMD_STAT + SIZE + (sum of all data bytes) and take the one's complement of the lowest byte of the sum. The one's complement was chosen so that the sum of zeros does not also get a checksum of zero.

e.g. for

RESET command: (0x10 + 0x00) XOR 0xFF = 0xEF

REMAP_RESET: (0x11 + 0x00) XOR 0xFF = 0xEE

6.4 Command list

Any valid command must have a value that is within the allowed range 0x10 ... 0xFF. Not all numbers are valid commands. The numbers 0x0 ... 0xF are reserved for status information and are forbidden for commands. See also Table 3 for further information about status values.

6.4.1 RESET

This command performs a software reset. It resets the CPU and also the digital core. It does not reset the analog blocks and the retention registers.

Reset is performed immediately without any delay.

Reset is done by performing an ARM system reset.

Information

Depending on the value of the remap bit in the CCU the system will start the bootloader again (if ROM is mapped to address 0) or an application in RAM (if there is one). If there is no application in RAM the system will have to be reset externally. I.e. by toggling the enable pin of the TOF chip or by power cycling.

Table 4: Reset command

Command	Value	Meaning
CMD_STAT	0x10	RESET
SIZE	0	No parameters
CSUM	0xEF	

6.4.1.1 Possible responses

- There is no STATUS set by the bootloader for this command in case the command was correct because a system reset is performed.
- If the command was incorrect the command will not be executed, and an error indication is set in CMD_STAT register.

6.4.2 RAMREMAP_RESET

This command remaps the RAM to address 0 and performs a system reset (see also command RESET).

The command is performed immediately without any delay.

After this the application that is stored in RAM will be running. If there is no valid application, you will need to do a hardware reset (toggle enable pin or power cycle).

Table 5: RAMREMAP command

Address	Value	Meaning
CMD_STAT	0x11	REMAP RAM to 0 and RESET
SIZE	0	No parameters
CSUM	0xEE	

6.4.2.1 Possible responses

- There is no STATUS set by the bootloader for this command in case the command was correct because a system reset is performed.
- If the command was incorrect the command will not be executed and an error indication is set in CMD_STAT register.

6.4.3 ROMREMAP_RESET

This command remaps ROM to address 0 and performs a system reset.

Table 6: ROMREMAP command

Address	Value	Meaning
CMD_STAT	0x12	REMAP RAM to 0 and RESET
SIZE	0	No parameters
CSUM	0xED	

6.4.3.1 Possible responses

- There is no STATUS set by the bootloader for this command in case the command was correct because a system reset is performed.
- If the command was incorrect the command will not be executed and an error indication is set in CMD_STAT register.

6.4.4 W_RAM

This command writes the given data to a defined RAM region. The RAM pointer has first to be set by the command ADDR_RAM. After the command W_RAM is successfully executed the RAM pointer will point to the first byte after the written region.

Table 7: W_RAM command

Address	Value	Meaning
CMD_STAT	0x41	Write to RAM
SIZE	0..0x80	Number of bytes to be written
DATA0	0..0xFF	1st byte to be written
DATA1	0..0xFF	2nd byte to be written
...		
DATA127	0..0xFF	128th byte to be written (only if size was 0x80)
CSUM	0..0xFF	The CSUM comes immediately after the data.

6.4.4.1 Possible responses

- If the command was correct READY is flagged in the CMD_STAT register.
- If the address is out of range the command will return ERR_RANGE.
- If the command was incorrect an error is flagged in CMD_STAT.

6.4.5 ADDR_RAM

This command is to specify the RAM pointer location for the next W_RAM command.

Figure 3: ADDR_RAM command

Address	Value	Meaning
CMD_STAT	0x43	Specify the address of the next RAM write.
SIZE	2	
DATA0	0..0xFF	LSB of Address in RAM
DATA1	0..0xFF	MSB of Address in RAM
CSUM	0..0xFF	

The bootloader will add the absolute RAM Base address to it. I.e. it will use the non-aliased address.

6.4.5.1 Possible responses

- If the command was correct READY is flagged in the CMD_STAT register.
- If the specified range was illegal an ERR_RANGE will be flagged.
- If the command was incorrect an error is flagged in CMD_STAT.

7 RAM patch download

Information

The TMF8806 can be operated directly from the on-chip ROM. A RAM patch download is usually not necessary. To run the ROM application, follow the procedure in chapter 6.

The Bootloader application allows you to download and start a firmware image.

To download an image, you need to perform the following steps:

1. Set up the RAM address pointer with the command ADDR_RAM.
2. Send consecutive data with the command W_RAM.
3. If you need to move the address pointer, use the command ADDR_RAM.
4. Send consecutive data with the command W_RAM.
5. Send the command RAMREMAP_RESET to start the downloaded application.

A simple example to download this Intel Hex snippet:

```
:020000042000DA
:100000006DC941853D15AA51F4D29EA8A7AC77E9E8
:10001000F9EC202463B8F1A50BA765B432B818D762
…
:101C1000FF8000D6EAF77C36807C00FF5D488E5D51
:04000005200000696E
:00000001FF
```

As I²C Strings:

Set up the address pointer, only the lower 16 address bits are used

(Intel Hex record :020000042000DA)

```
S 41 W 08 43 02 00 00 BA P
```

Send first data record

(Intel Hex record :100000006DC941853D15AA51F4D29EA8A7AC77E9E8)

```
S 41 W 08 41 10 6D C9 41 85 3D 15 AA 51 F4 D2 9E A8 A7 AC 77 E9 A6 P
```

Read back status register (should read back as 00 00 FF)

```
S 41 W 08 Sr 41 R A A N P
```

Send 2nd data record which is in this case consecutive as it starts at address 0x10

(Intel Hex record :10001000F9EC202463B8F1A50BA765B432B818D762)

```
S 41 W 08 41 10 F9 EC 20 24 63 B8 F1 A5 0B A7 65 B4 32 B8 18 D7 30 P
```

Read back status register (should read back as 00 00 FF)

```
S 41 W 08 Sr 41 R A A N P
```

…

Read back Status register (should read back as 00 00 FF)

```
S 41 W 08 Sr 41 R A A N P
```

Set up address pointer to address 0x2000_1C10 (Intel Hex record :101C1000FF8000D6EAF77C36807C00FF5D488E5D51) – This is only necessary if the image was not continuous at this point.

```
S 41 W 08 43 02 10 1C 8E P
```

Read back Status register (should read back as 00 00 FF)

```
S 41 W 08 Sr 41 R A A N P
```

Send the data of the above Intel Hex record.

```
S 41 W 08 41 10 FF 80 00 D6 EA F7 7C 36 80 7C 00 FF 5D 48 8E 5D 3B P
```

Read back Status register (should read back as 00 00 FF)

```
S 41 W 08 Sr 41 R A A N P
```

Ignore the Intel Hex Record Type 05 = Start Linear Address.

The Intel Hex Record Type 01 = EOF triggers the sending of the RAMREMAP_RESET command:

```
S 41 W 08 11 00 EE P
```

After this command is sent, the host should poll the register 0xE0 for CPU Ready or wait for a timeout.

8 App0

The commands described in this chapter only work if the system ran through the general flow described in chapter 2 until App0 firmware startup.

8.1 Command overview

As soon as the application firmware (App0) is running the following I²C commands can be used for communication with the application:

1. Factory calibration
2. Read out App0 version, hardware version and the serial number of the hardware
3. Set factory calibration data
4. Start App0
5. Read back results
6. Stop App0
7. Readout of raw histograms
8. Oscillator re-trimming

8.2 Check command execution status

You can check if an issued command was processed successfully with this sequence:

1. Send command
2. Read register 0x10 and 0x11
3. Check if register 0x10 equals 0x00 and register 0x11 equals the command code sent in step 1. In this case command execution was successful. Exit with status OK
4. Read register 0x1C (state register, see 12.2)
5. Exit with an error if register 0x1C equals 0x02 (error state)
6. Repeat from step 2 until time-out

8.3 Interrupts

App0 supports two interrupts to signal events to the host:

- Result interrupts to signal measurement results and factory calibration data.
- Diagnostic interrupts for histogram readout.

Bit0 in register 0xE1 will be set if **result** interrupts are triggered by the firmware. If bit0 in register 0xE2 is set the interrupt pin will be asserted in addition.

Bit1 in register 0xE1 will be set if **diagnostic** interrupts are triggered by the firmware. If bit1 in register 0xE2 is set the interrupt pin will be asserted in addition.

Writing to these bits after an interrupt is triggered will clear the corresponding interrupt (self set, write clear).

8.4 Factory calibration

The device must be factory calibrated to be able to report the correct distances in the final environment.

Calibration environment:

- Device has to be in the final (correct) optical stack
- Cover glass (no smudge on the glass) in front of the device
- No target in front of the device within 40cm (see datasheet)
- Dark room or low ambient light

The factory calibration sequence is started by issuing a device configuration with command 0xA to the App0. This example is with the default configuration. The calibration iterations are 0xA000 kIters = 40.96 MIters. Have a look at the datasheet for details on the configuration.

```
S 41 W 06 00 00 00 02 00 00 00 64 00 A0 0A P
```

After this factory calibration started. You can wait for the result interrupt to get triggered, or poll the register 0x1E until it reads back as 0x0A:

```
S 41 W 1E Sr 41 R A N P
```

The factory calibration time depends on the factory calibration confioguration and iterations.

Now read back and store the 14 bytes of factory calibration.

```
S 41 W 20 Sr 41 R A A A A A A A A A A A A A N P
```

e.g. one device reads back this data:

```
S 41 W 20 Sr 41 R 02 00 00 12 70 FE 01 04 07 08 36 24 00 04 P
```

This information must be written to the TMF8X0X every time after a power cycle. See chapter 8.6.

Attention

The factory calibration also calibrates the optical zero-position of the coverglass.
To do that, device configurations like SPAD settings, distance modes, or VCSEL clock rate of the factory calibration must match the target application.
For factory calibration it is recommended to select a high number of iterations (like 40 million iterations).

8.5 App0 version

To retrieve the App0 version information send the following I²C strings:

```
S 41 W 01 Sr 41 R N P
```

Returns the major version number of the App0.

```
S 41 W 12 Sr 41 R A N P
```

Returns the minor and patch revision number of the App0.

```
S 41 W E3 Sr 41 R A N P
```

Returns the ID and REVID of the chip (HW version).

To read out the serial number the command SERIAL NUMBER must be written and then the number can be read back.

```
S 41 W 10 47 P
```

Read back register 0x1E until it has the value 0x47 (serial number).

```
S 41 W 1E Sr 41 R N P
```

Read back the serial number:

```
S 41 W 28 Sr 41 R A A A N P
```

8.6 Set factory calibration data

This example downloads the factory calibration data to the App0.

Calibration Data example (you must use the data you stored from the factory calibration – see Chapter 8.4):

```
S 41 W 20 02 00 00 12 70 FE 01 04 07 08 36 24 00 04 P
```

Factory calibration data is only preloaded into the I²C registers here and will be processed after measurements have been started by the host.

8.7 Set state data

The TMF8806 App0 firmware always appends algorithm state data to each measurement result frame (see data sheet description of register STATE_DATA_0 and following). The host can cache this data and provide it to the firmware when restarting measurements.

This is required especially for ultra low operation, where the algorithm state is transferred from the host to TMF8806 right after startup.

This example downloads the state data to the App0:

```
S 41 W 2e b1 a9 02 00 00 00 00 00 00 00 00 P
```

8.8 Start measurements

Use the factory calibration data from chapter 8.4. Run measurements in continuous mode, period of 33 ms, 900k iterations, GPIOs are not used.

```
S 41 W 06 00 00 11 02 00 00 06 1E 84 03 02 P
```

Explanation (see datasheet for details):

cmd_data9=00	No spread spectrum for SPAD charge pump
cmd_data8=00	No spread spectrum for the VCSEL charge pump
cmd_data7=11	Factory calibration is provided, SPAD dead time 4
cmd_data6=02	Enable distance algorithm (in addtition to proximity)
cmd_data5=00	No GPIO control used
cmd_data4=00	No GPIO control used
cmd_data3=06	Object detection threshold, use 6 as default
cmd_data2=1E	Repetition period in milliseconds 0x1E hex = 33 ms
cmd_data1=84	Number of kilo-iterations, low byte
cmd_data0=03	Number of kilo-iterations, high byte. 0x0384 = 900k iterations.
command executed=02	Run measurements

If you preloaded state data (see chapter 8.7) set register cmd_data7 = 0x3 indicating that factory calibration data and state data were provided to start the measurements.

8.9 Read results

S 41 W 1D Sr 41 R A A A A A A A A A A N P

Register 0x1D reports the status of TMF8806 (detailed explanation in section 12.2).

If the content of register 0x1E is 0x55 the registers 0x20 and following contain the distance measurement result. You can also wait for the result interrupt.

The host should check that the number in register 0x20 increases every time a new result is calculated. See also the datasheet for interpretation of the registers.

A new result will be reported by the TMF8806 every 100ms in this example. Please note that the host has to compensate for oscillator drift between host and TMF8806.

Attention

Always start reading from 0x1D with a bulk read to correctly read registers SYS_CLOCK_x. See datasheet for details.

Information

If state data needs to be readout, simply continue reading after registers SYS_CLOCK_x. To allow resume of operation after power-off, algorithm state can be stored temporarily inside the host and once after power-on of TMF8806 restored to resume operation.

8.10 Stop measurements

```
S 41 W 10 FF P
```

After issuing the command, the application will terminate measurements as quickly as possible. This time depends on the internal sensor state, and is 4.5ms in worst case.

8.11 Readout of raw histograms

In order to readout the raw histogramd, the TMF8806 needs to be configured in this mode. The procedure is as follows:

Stop any command:

```
S 41 W 10 FF P
```

Check if STOP command was successful (see chapter 8.2)

```
S 41 W 11 Sr 41 R N P
```

```
Clear INT_STATUS
```

```
S 41 W E1 01 P
```

```
Read back INT_STATUS
```

```
S 41 W E1 Sr 41 R N P
```

Configure to read out short range histogram (example for or – see datasheet for available histograms)

```
S 41 W 0C 10 00 00 00 30 P
```

Configure for cyclic measurement. Default configuration, no calibration data no state data.

```
S 41 W 06 00 00 00 02 00 00 00 64 84 03 02 P
```

Check if command was successful (see chapter 9.2)

Loop begin here:

```
Wait for INT_STATUS to have bit 1 (0x02) set
```

```
S 41 W E1 Sr 41 R N P
```

… repeat as needed until INT_STATUS gets 0x02 …

```
S 41 W E1 Sr 41 R N P
```

Read out COMMAND, PREV_COMMAND ... STATE STATUS REGISTER_CONTENTS -> store TID

```
S 41 W 10 Sr 41 R A A A A A A A A   A A A A A A A A P
```

Start to read out raw histogram

```
S 41 W 10 80 P
```

Read out TID register - wait for TID to change (compare with TID read above)

```
S 41 W 1f Sr 41 R A P
```

Once TID did change continue and read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Attention

The actual readout of the histogram itself shall be done with a single I²C blockread (128 bytes).
If the platform does not support such large I²C block reads, the host can split it into smaller chunks. Then the host must ensure that the block containing register 0x30 is read as last block of the histogram.

Read out TDC0 quater 0: Bin 0..63: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC0 quater 1: Bin 64..127: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC0 quater 2: Bin 128..195: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC0 quater 3: Bin 196..255: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC1 quater 0: Bin 0..63: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC1 quater 1: Bin 64..127: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A P

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

S 41 W 1c Sr 41 R A A A N P

Read out TDC1 quater 2: Bin 128..195: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A P

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

S 41 W 1c Sr 41 R A A A N P

Read out TDC1 quater 3: Bin 196..255: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A P

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

S 41 W 1c Sr 41 R A A A N P

Read out TDC2 quater 0: Bin 0..63: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A P

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

S 41 W 1c Sr 41 R A A A N P

Read out TDC2 quater 1: Bin 64..127: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A P

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

S 41 W 1c Sr 41 R A A A N P

Read out TDC2 quater 2: Bin 128..195: LSB+MSB

S 41 W 20 Sr 41 R A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A

```
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC2 quater 3: Bin 196..255: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC3 quater 0: Bin 0..63: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC3 quater 1: Bin 64..127: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC3 quater 2: Bin 128..195: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC3 quater 3: Bin 196..255: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC4 quater 0: Bin 0..63: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC4 quater 1: Bin 64..127: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC4 quater 2: Bin 128..195: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Read out STATE STATUS REGISTER_CONTENTS TID - wait for TID to change

```
S 41 W 1c Sr 41 R A A A N P
```

Read out TDC4 quater 3: Bin 196..255: LSB+MSB

```
S 41 W 20 Sr 41 R A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A
A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A
A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  A A A A A
A A A  A A A A A A A A  A A A A A A A A  A A A A A A A A  P
```

Clear diagnostic interrupt

S 41 W E1 02 P

Continue normal operation

S 41 W 10 32 P

Poll INT_STATUS register for bit 0 (0x01) to be set (or wait for the interrupt to be asserted)

```
S 41 W E1 Sr 41 R N P
```

Clear INT_STATUS

S 41 W E1 01 P

Read out STATE STATUS REGISTER_CONTENTS TID RESULT_NUMBER ...

```
S 41 W 1c Sr 41 R A A A A   A A A A A A A A P
```

Continue with loop above (=next measurement).

8.12 Additional configuration for App0 / interrupt suppression

There are two more commands to set and get the additional configuration parameters. These additional parameters are: persistence, low threshold and high threshold.

The purpose is to limit the number of interrupts that are generated by the TOF device. With tehe default configuration (when persistence is 0), the device will generate a result interrupt for every measurement period, no matter if there was an object or not.

By setting persistence to 1 the device will only produce an interrupt if there was an object in the specified range [low threshold ... high threshold].

By setting persistence to 2 the device will only produce an interrupt if there was an object in the specified range [low threshold ... high threshold] for two consecutive measurements.

By setting persistence to n the device will only produce an interrupt if there was an object in the specified range [low threshold ... high threshold] for n consecutive measurements.

Note that a persistence of e.g. 3 means that for an initial report of the object there need to be 3 measurements in sequence that have an object in the specified range. After this initial phase each period an interrupt will be generated as long as the object is in the specified range.

The 3 additional configuration parameters are:

- Persistence = Unsigned byte (0 … 255)
- Low Threshold [mm] = Unsigned short (0 ... 65535)
- High Threshold [mm] = Unsigned short (0 ...65535)

Information

If the low threshold is set to a higher value than the high threshold, no object can be reported.

8.12.1 Write persistence and thresholds WR_ADD_CONFIG (0x08)

The command follows the structure of the other commands – i.e. the parameters are written to the registers cmd_data* before writing the command register itself.

Please check the data sheet for more information.

e.g. for a persistence of 5, low threshold of 55 mm and high threshold of 500mm (55 = 0x37, 500 = 0x1F4):

```
S 41 W 0b 05 37 00 f4 01 08 P
```

8.12.2 Read persistence and thresholds RD_ADD_CONFIG (0x09)

This command has no parameters. Please check the data sheet for more information.

e.g. to read back the values from above:

```
S 41 W 10 09 P
```

S 41 W 11 Sr 41 R N P -> wait until register 0x11 reads back as 09

Read out the parameters:

```
S 41 W 1E Sr 41 R A A A A A A N P -> 09 <tid> 05 37 00 f4 01
```

9 Timings

9.1 Startup timings

All timings in the table below are given starting at 0 ms when the enable line is pulled high. If two lines show the same timestamp, there is no wait time required between these two lines. The execution should follow the order given in the table. This example starts the TMF8806 ROM application and performs a measurement.

Table 8: Timings

Time stamp	I²C string	Description
0 ms	Enable power line (set HIGH)	Power TMF8806, hardware starts, bootloader puts TMF8806 into power down mode
1.6 ms	Do not send any I²C commands	I²C slave is available (Registers 0xE0 … 0xFF only)
2.5 ms	S 41 W E0 01 P	Set PON=1, wake up bootloader
3.6 ms	S 41 W E0 Sr 41 R N P	bootloader is ready (read register 0xE0 as 0x41)
3.7 ms	S 41 W 02 C0 P	Start ROM application
4.4 ms	S 41 W 00 Sr 41 R C0 P	Application is ready

Time stamp	I²C string	Description
4.5 ms	S 41 W E1 01 P S 41 W E2 01 P	Enable and clear interrupts for results
4.7 ms	S 41 W 20 <CALIB><STATE> P	Set factory calibration (and state for hibernate)
5.0 ms	S 41 W 068 41 10 4D B9 42 …66 <CONFIG> P	Send configuration and start measurement (or calibration)
5.6 ms	S 41 W 08 Sr 41 00 02 P	Measurement started (calibration reads back 00 0A)
X.0 ms	INT-Pin pulled low	Result is ready
X.0 ms	S 41 W E1 Sr 41 R 01 P	Check result interrupt
X.1 ms	S 41 W E1 01 P	Clear interrupts
X.1 ms	S 41 W 1C Sr 41 R RES> P	Read results and status

Information

If you send a command W_RAM with 16 bytes you must wait for 150 microseconds for the status to become ready. If you send a command W_RAM with the maximum payload of 128 bytes you must wait for 1 millisecond for the status to become ready.

9.2 Measurement result timings

After the start of the measurement, an interrupt is triggered after the configured period in milliseconds. The result data is available when the interrupt occurs.

There are reasons the period cannot be met:

- The oscillator drift has to be taken into consideration so the interrupt can occur ± 4% later/ earlier.
- If the integration + distance algorithm computation time exceeds the measurement period, the measurement period will be delayed.
- If the temperature changes over time, the device needs to internally re-calibrate. This re-calibration takes a few milliseconds and can delay a measurement.

After issuing the Stop command, the application will terminate as quickly as possible. This time depends on the internal state. In the worst case a stop command takes 4.5ms to take effect.

10 Oscillator drift correction

The internal oscillator of the TMF8806 can drift over time due to e.g., temperature changes. This drift has a negative impact on the measurement accuracy.

The TMF8806 application firmware generates a timestamp as part of the measurement result structure.

The host can compare the result timestamp difference against a precise reference clock (e.g., the host system tick). The host can then compensate for the measurement distance error caused by the internal oscillator drift.

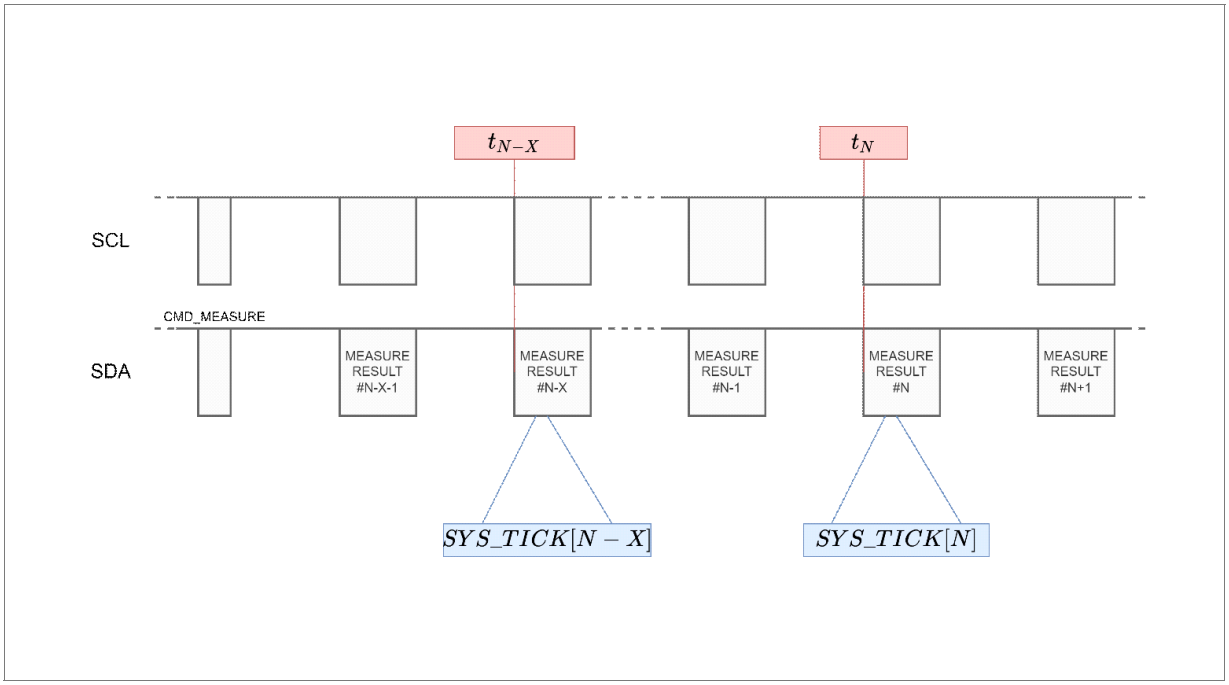
The TMF8806 reports its internal timestamp since it has been activated (App0 and PON=1). The resolution of the timestamp is 1 / 4.7MHz = 0.21277 microseconds. The reported value is an unsigned 32-bit value that wraps around approximately every 15 minutes.

The timestamp is reported by the TMF8806 in registers SYS_CLOCK_0, SYS_CLOCK_1, SYS_CLOCK_2, SYS_CLOCK_3 in little endian format. To get this information you must perform an I²C block read starting from address 0x1D (see also the register description section of the datasheet).

If the LSB of the timestamp is 0, the timestamp value is invalid and should not be considered for oscillator drift correction.

10.1 Calculate and compensate the oscillator drift

Figure 4: Read the timestamp to calculate the oscillator drift



The host can compute the oscillator drift correction by comparing two TMF8806 timestamps with timestamps of the I²C transfers:

Equation 1:

$$correction\_factor[t_N] = \frac{t_N - t_{N-X}}{(\text{SYS\_TICK}[N] - \text{SYS\_TICK}[N-X]) * 212{,}77\text{ns}}$$

The host can then correct the reported distance by the drift correction:

Equation 2:

$$actual\_distance[t_N] = reported\_distance[t_N] \cdot correction\_factor[t_N]$$

Information

The host should use two timestamps that are far apart for the correction factor to reduce the jitter. For example, the host can use a delay of 16 results at a measurement period of 33ms to compute the current correction factor, resulting in a correction time of roughly 528ms.

11 Oscillator re-trimming

If oscillator re-trimming is required, the following procedure must be done by the host driver.

1. Stop a running application
2. Write the software password 0x29 to I²C register 0x06
3. Put the device into standby mode (write the value 0x00 to register 0xE0)
4. Wait for the device to enter standby mode (read register 0xE0 until it reads back as 0x00).
5. Read the content of register 0x03 and register 0x06. The 8 bits of register 0x03 plus bit 6 of register 0x06 together are the 9-bit signed value – here called the osc_trim_value. (See

section 11.2 how to build this value.)

6. Adjust the osc_trim_value. Split the 9-bit signed value into a single LSB bit and an 8-bit value.
7. Write the LSB of the osc_trim value to bit 6 of register 0x06
8. Write the upper 8 bits of the osc_trim value to register 0x03
9. Wake up the device from standby state (write the value 0x01 to register 0xE0).
10. After this procedure the device will operate with the changed frequency.

Attention

Do not change any other bit than bit6 in register 0x06 or any of the other registers except for register 0x03, in this special standby mode. If you do the firmware will automatically reset all registers to the original fuse values and the oscillator re-trim will be discarded.

Information

Re-trimming the oscillator outside the allowed range of 4.5MHz – 5.5MHz can lead to a lockup of the device.
It is recommended to perform small steps (range: ±1... ±8).

Information

Running the TMF8806 above 4.75MHz will reduce maximum measurement distance.

11.1 Re-trimming example

Step 1

Enable the ToF chip and download the latest application firmware.

Step 2

After download the application firmware is in idle state (no measurement running). No need to stop it.

Write the software password:

S 41 W 06 29 P

Put the device into standby mode:

S 41 W E0 00 P

Check that the device is in standby mode (register 0xE0 == 0x00), device response after “->”

S 41 W E0 Sr 41 R A P -> S 41 W E0 Sr 41 R 00 P

Step 3

Now modify the trim value. Read registers 0x03 to 0x06, device response after “->”

S 41 W 03 Sr 41 R A A A A P -> S 41 W 03 Sr 41 R 1a 0c 1c 40 P

Modify the trim value in register 0x03 (decrease by one):

S 41 W 03 19 P

Wake up the device from standby:

S 41 W E0 01 P

Check that device woke up (register 0xE0 == 0x41), device response after “->”

S 41 W E0 Sr 41 R A P -> S 41 W E0 Sr 41 R 41 P

Step 4

Now check if the device kept the trim value.

Write the software password:

S 41 W 06 29 P

Put the device into standby mode:

S 41 W E0 00 P

Check that the device is in standby mode (register 0xE0 == 0x00), device response after “->”

S 41 W E0 Sr 41 R A P -> S 41 W E0 Sr 41 R 00 P

Read registers 0x03 to 0x06, device response after “->”

S 41 W 03 Sr 41 R A A A A P -> S 41 W 03 Sr 41 R 19 0c 1c 40 P

Register 0x03 kept the value 0x19 after re-trimming. That is the expected behavior.

11.2 How to build the osc_trim_value:

The functions RReg8(addr) assumes that it will read the register value from the specified address via I²C.

The function WReg8(addr,value) assumes that it will write the value to the specified register via I²C.

Information

For the code below to work, you need to make sure that the shifting operation in line 4 below does a sign-extend. Usually, it should do a sign extend, if you use for the variable the machine word type (e.g. on a 32-bit processor it should be a signed integer of 32 bits).

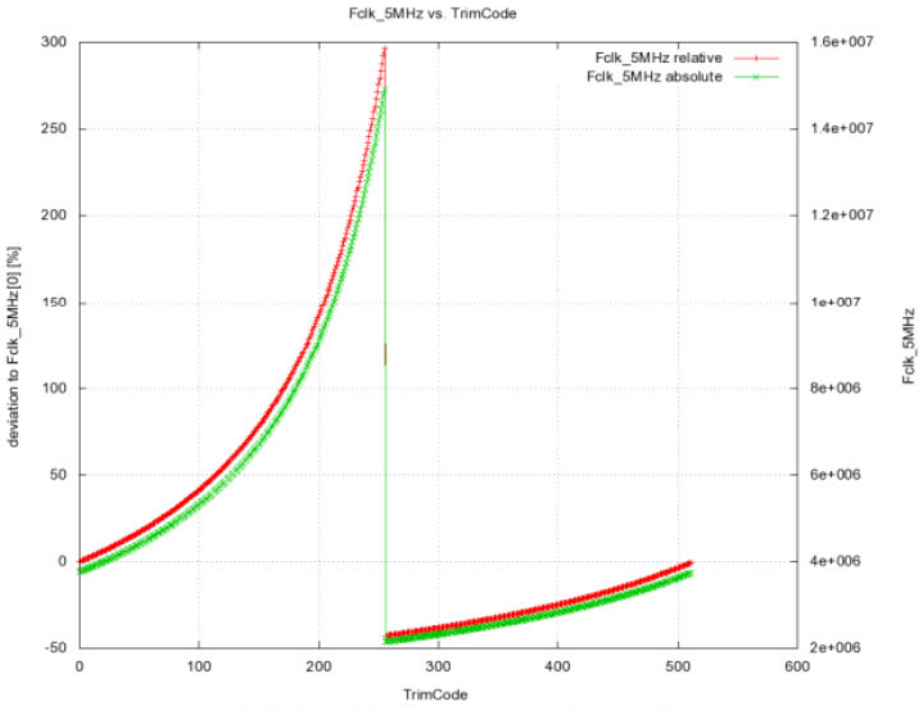
Figure 5: Code example for oscillator re-trimming

```
int32_t osc_trim_value; /* must be machine word type */
uint8_t RReg8( uint8_t addr );
void WReg8( uint8_t addr, uint8_t value );

uint8_t reg6Value = RReg8( 0x06 );
osc_trim_value = RReg8( 0x03 );
/* construct 9-bit unsigned value */
osc_trim_value = (osc_trim_value << 1 ) | ( ( reg6Value >> 6 ) & 1 );
osc_trim_value = (osc_trim_value << 23 ) >> 23; /* sign extend */
osc_trim_value += stepUpDown;
reg6Value = reg6Value & ~( 1 << 6 );  /* clear bit 6 */
/* write the LSB only */
reg6Value = reg6Value | ( ( osc_trim_value & 1 ) << 6 );
/* write upper 8 bits (incl. sign bit) */
WReg8( 0x03, ( osc_trim_value >> 1 ) );
WReg8( 0x06, reg6Value );
```

11.3 Nominal oscillator trim values

Figure 6: Nominal oscillator trim values



Plot 37: oscillator frequency range & availability

Information

The trim value is a signed 9-bit value, the picture above shows it as an unsigned 9-bit value. The unsigned 9-bit values [256 .. 511] correspond to [-1 .. -256] 9-bit signed values.

The picture above shows the nominal trim curve. Note that every device will be trimmed to the value that is correct for this device. So there is no absolute value for 5MHz.

The device is trimmed correctly during production test.

Adjusting the trim-value always must be a done as a read-modify-write step.

12 Appendix

12.1 I²C slave address change

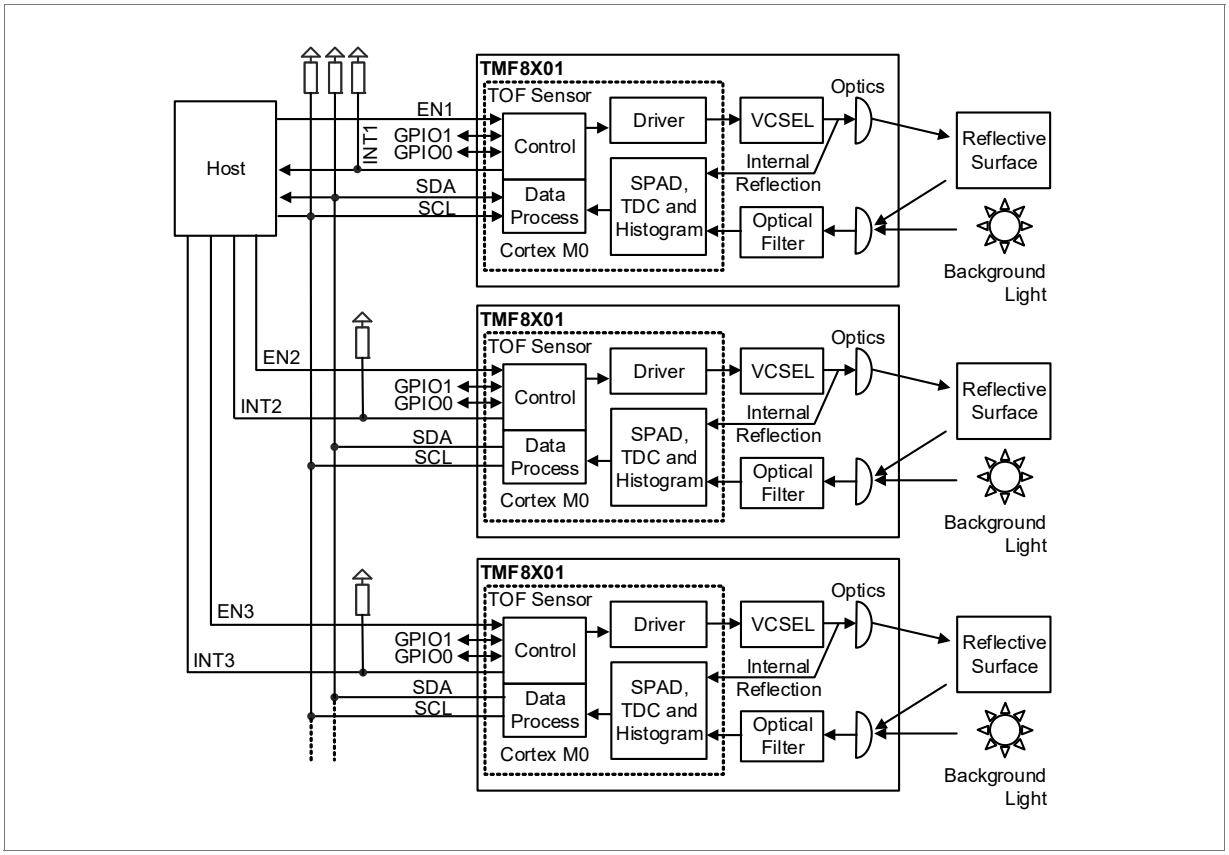
The I²C address of the ToF device can be dynamically changed inside a running application.

You can do this by using individual enable lines to each device. You need one individually controllable GPIO line for each ToF device on the same I²C bus.

12.1.1 Enable line controlled I²C address change

This section describes how to perform the address change when having one individually controlled enable line per ToF device. I.e. the host driver must be capable of driving each GPIO line high or low.

Figure 7: Enable line control schematic for dynamic I²C address change



In the example below, there are four ToF devices that are being reprogrammed to use different addresses. The host driver drives all enable lines low. All ToF devices are in Power Off state.

1. The host driver drives a single enable line high. The ToF device connected to this enable line will enter the power down state.
2. The host driver sends the following I²C string:
   S 41 W E0 01 P
3. The host driver reads back until the I²C register 0xE0 changes to the value 0x41
4. The host driver downloads the patch through I²C bootloader commands.
5. The host driver tells the bootloader to perform a RAMREMAP_RESET command.

Now the ToF device is ready to be reprogrammed.

6. The host driver sends the following I²C string (assuming the device shall be reprogrammed to 7-bit address 0x51 == upshifted by 1 to 0xA2):
   S 41 W 0E A2 00 49 P
7. The host driver sends the following I²C string (this I²C request might actually fail, if the device has itself already reprogrammed to the new address – this depends on the internal state of the device).
   S 41 W 10 ff P
8. Now the device is reprogrammed. You can test to access it by sending the following string to read out register 0xE0 (the device will have the value 0x41 in this register):
   S 51 W e0 Sr 51 R A P

To reprogram another device, repeat steps 1., 2., 3., 4., 5., 6., exactly like above and steps 7. and 8. with a new address e.g. 0x52. So use for those two the following 2 strings:

S 41 W 0e a4 00 49 P

and

S 41 W 10 ff P

And to see that the new device is there, read back with:

S 52 W e0 Sr 52 R A P

Repeat the above for as many devices as you have on this I²C bus. Make sure each gets its own unique I²C slave address on the I²C bus.

12.2 State register (Address 0x1C)

The host driver checks register 0x1C during command handling. Please see chapter 8.2.

12.3 Status register (Address 0x1D)

Register 0x1D gives the internal status of the application running on TMF8806. Note that the status register is only valid for readout if the state field indicates a permanent state. I.e. you should only interpret the value of register 0x1D when register 0x1C has the value 1, 2 or values 17-28 in case you are using diagnostic states.

TMF8806 updates the status register every time the internal state machine traverses through states. Only errors are sticky and remain.

Table 9: Status register codes

Status	Value	Description
Idle	0x00	No error, information that internal state machine is idling.
Diagnostic	0x01	No error, information that internal state machine is in diagnostic mode.
Start	0x02	No error, internal state machine is in initialization phase.
Calibration	0x03	No error, internal state machine is in the calibration phase (breakdown voltage, electrical calibration or optical calibration).
LightCol	0x04	No error, internal state machine is performing HW measurements and running the proximity algorithm.
Algorithm	0x05	No error, internal state machine is running the distance algorithm
Startup	0x06	No error, internal state machine is initializing HW and SW.
VcselPwrFail	0x10	Error, eye safety check failed, VCSEL is disabled by HW circuit.
VcselLedAFail	0x11	Error, eye safety check failed for anode. VCSEL is disabled by HW circuit.
VcselLedKFail	0x12	Error, eye safety check failed for cathode. VCSEL is disabled by HW circuit.
CalibError	0x1b	Error, electrical calibration failed. No two peaks found to calibrate.
InvalCmd	0x1c	Error, either a command byte was written to register 0x10 that is not supported by the application, or the command was sent while the application was busy executing the previous command. I.e. no stop command was sent before.
ErrMissingFactCal	0x27	There is no (or no valid) factory calibration on the device. Using default values instead.
ErrInvalidFactCal	0x28	Parsing of the provided factory calibration found an illegal calibration value.

Status	Value	Description
ErrInvalidAlgState	0x29	Parsing of the provided algorithm state found an illegal algorithm state value.

13 Revision Information

Changes to current revision v1-00	Page
Initial production version	

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

ABOUT ams OSRAM Group (SIX: AMS)

The ams OSRAM Group (SIX: AMS) is a global leader in optical solutions. By adding intelligence to light and passion to innovation, we enrich people's lives. This is what we mean by Sensing is Life. With over 110 years of combined history, our core is defined by imagination, deep engineering expertise and the ability to provide global industrial capacity in sensor and light technologies. Our around 24,000 employees worldwide focus on innovation across sensing, illumination and visualization to make journeys safer, medical diagnosis more accurate and daily moments in communication a richer experience. Headquartered in Premstaetten/Graz (Austria) with a co-headquarters in Munich (Germany), the group achieved over EUR 5 billion revenues in 2021. Find out more about us on https://ams-osram.com

DISCLAIMER

PLEASE CAREFULLY READ THE BELOW TERMS AND CONDITIONS BEFORE USING THE INFORMATION SHOWN HEREIN. IF YOU DO NOT AGREE WITH ANY OF THESE TERMS AND CONDITIONS, DO NOT USE THE INFORMATION.

The information provided in this general information document was formulated using the utmost care; however, it is provided by ams-OSRAM AG or its Affiliates* on an "as is" basis. Thus, ams-OSRAM AG or its Affiliates* does not expressly or implicitly assume any warranty or liability whatsoever in relation to this information, including – but not limited to – warranties for correctness, completeness, marketability, fitness for any specific purpose, title, or non-infringement of rights. In no event shall ams-OSRAM AG or its Affiliates* be liable – regardless of the legal theory – for any direct, indirect, special, incidental, exemplary, consequential, or punitive damages arising from the use of this information. This limitation shall apply even if ams-OSRAM AG or its Affiliates* has been advised of possible damages. As some jurisdictions do not allow the exclusion of certain warranties or limitations of liabilities, the above limitations and exclusions might not apply. In such cases, the liability of ams-OSRAM AG or its Affiliates* is limited to the greatest extent permitted in law.

ams-OSRAM AG or its Affiliates* may change the provided information at any time without giving notice to users and is not obliged to provide any maintenance or support related to the provided information. The provided information is based on special conditions, which means that the possibility of changes cannot be precluded.

Any rights not expressly granted herein are reserved. Other than the right to use the information provided in this document, no other rights are granted nor shall any obligations requiring the granting of further rights be inferred. Any and all rights and licenses regarding patents and patent applications are expressly excluded.

It is prohibited to reproduce, transfer, distribute, or store all or part of the content of this document in any form without the prior

For further information on our products please see the Product Selector and scan this QR Code.

Published by ams-OSRAM AG
Tobelbader Strasse 30, 8141 Premstaetten, Austria

Published by ams-OSRAM AG
Tobelbader Strasse 30,
8141 Premstaetten Austria
Phone +43 3136 500-0
ams-osram.com