Product
Document

am **OSRAM**

Application Note

AN000597

TMF8X0X Host Driver Communication

I²C Commands to Control TMF8701/TMF8801/TMF8805

v8-00 • 2022-Nov-30

Content Guide

1   Introduction

This document describes the I²C communication between TMF8701/TMF8801/TMF8805 (=TMF8X0X) and an I²C host.

Please refer to the TMF8X0X datasheets 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 TMF8X0X 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.

1.1   Ordering Information

Ordering Code	Description
TMF8701-EVM	TMF8701 ToF Evaluation Module
TMF8701-DB	TMF8701 ToF Daughter Board
TMF8801-EVM	TMF8801 ToF Evaluation Module
TMF8801-DB	TMF8801 ToF Daughter Board
TMF8805-EVM	TMF8805 ToF Evaluation Module
TMF8805-DB	TMF8805 ToF Daughter Board

2 General Flow

1. Power the TMF8X0X (VDD)

2. Pull Enable Line High

3. Write a 0x1 to register 0xE0 (ENABLE) (see section 3)

4. Poll register 0xE0 until the value 0x41 is read back (see section 4.1)

5. Read the register 0x00 (APP_ID) to find out what application is running (see section 5.1)

6. If the bootloader (register 0x00 == 0x80) is running download a RAM patch (see section 6)

7. Start the RAM patch (RAMREMAP_RESET command – section 6.4.2)

8. Poll register 0xE0 until the value 0x41 is read back (see section 4.1)

9. Read the register 0x00 to find out if App0 is running (see section 5.1) - read back value should be 0xC0 for App0

10. Talk to the App0 (see section 8)

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 ToF device and also how the host should communicate with the ToF during and immediately after the startup.

3.1   ToF Startup Behavior

TMF8X0X has 2 different applications built into ROM.

- Bootloader Application (I²C Register 0 has the value 0x80)
- App0 Application (I²C Register 0 has the value 0xC0), outdated, so do not use without patching.

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

Figure 1:
Bootloader Startup Behavior
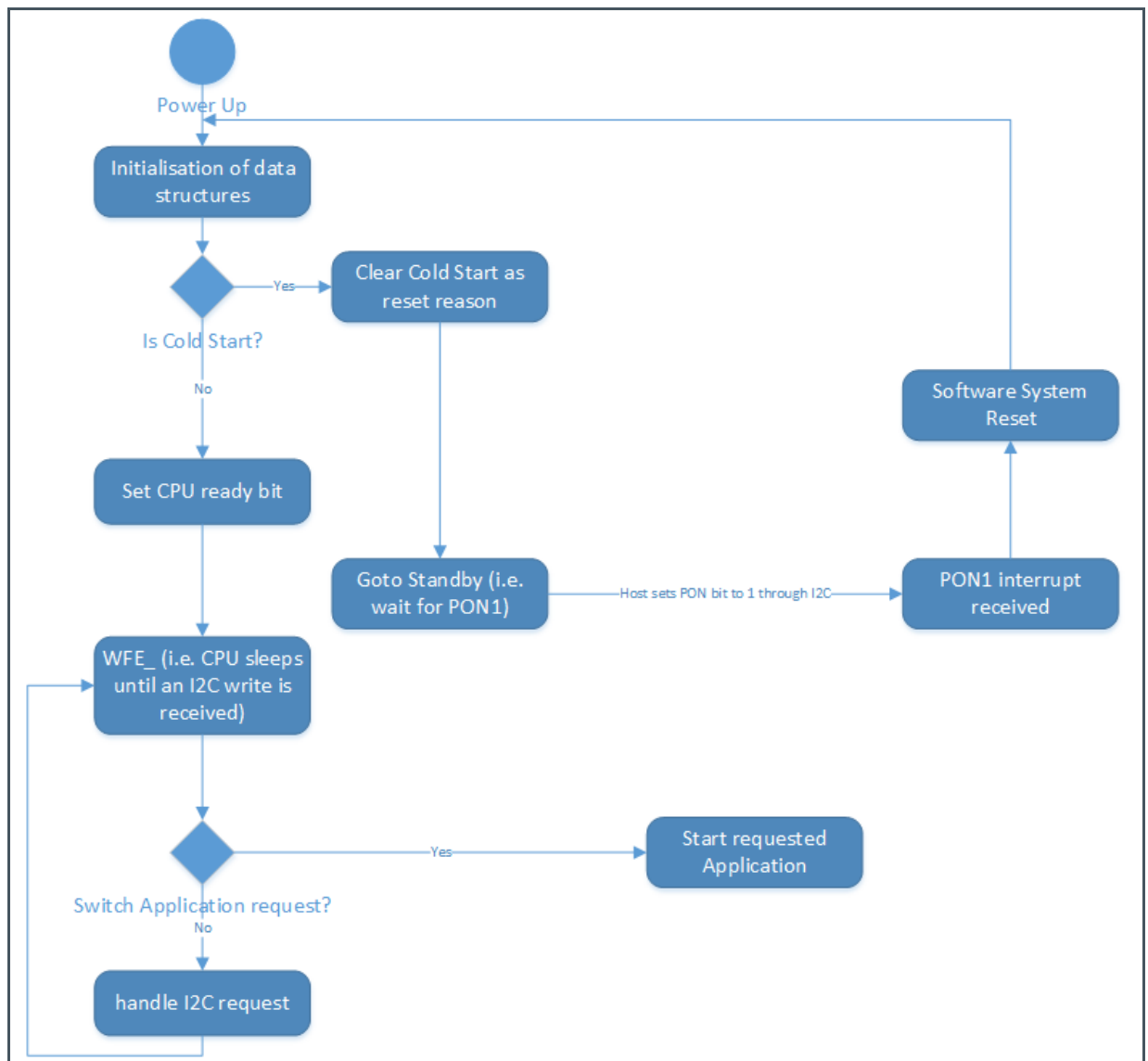
The bootloader application keeps running and waits for inputs from a host via I²C.

3.2 Host Behavior at and After ToF Startup

The host can control the ToF device through the following lines: Power-line, enable-line, SDA, SCL

The device can inform the host through the following line: Interrupt

If the host powers the ToF device and sets the enable line high, the ToF will perform startup 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
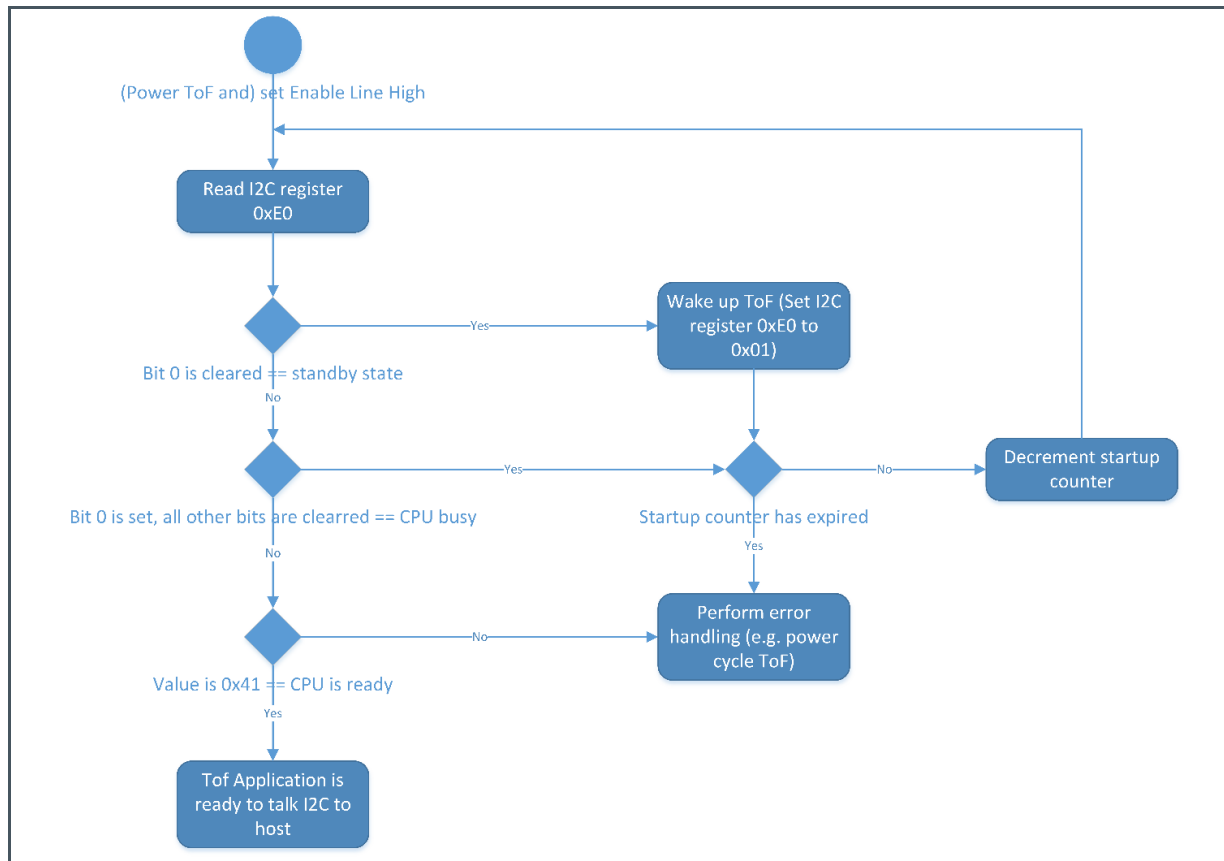
Note that the above figure is only true if the ToF has just been powered up or the enable line has been just set high. Later the ToF may enter e.g. the deep sleep state and the register 0xE0 can have different values (see also TMF8X0X datasheets for further details).

To set pon=1 send following I²C sequence:

S 41 W E0 01 P

To read register 0xE0 send the following I²C sequence:

S 41 W E0 Sr 41 R N P

4   ToF State – At Any Time

To find out the state of the ToF or to put the ToF 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 ToF is undefined, and the read results are unreliable

4.1   Is 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 SW has not yet set the cpu_ready bit.
- If the read value is 0x00 the ToF is in standby state and can only be waked up by the sequence given in section 4.2
- If you read any other value check the TMF8X0X datasheets

> **(i) 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.
>
> **Note for host developer:** The host should only talk to the I²C SW registers 0x00 – 0xDF when the CPU ready bit is set and the ToF is neither in standby nor in deep sleep state. All ToF applications should clear the CPU ready bit before entering standby state or deep sleep state, however it is better if the host checks also that the ToF is in operational mode. As all these states are represented in the same I²C register (0xE0) this is considered acceptable. The application is operational only if register 0xE0 reads back as 0x41.

4.2 Wake Up from Standby State

To wake up the ToF 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 ToF device and the ISR for PON1 will perform a SW reset and after that the CPU should get 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 goto standby function. This function shuts down the PLL, CP, etc. and enables the PON1 interrupt.

5   Talk to ToF

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 ToF. The ToF will only change the content of the 0xE0 register when requested to do so by the host – or when an unforeseen error occurs.

5.1   Find Out Which Application Is Running

The I²C SW 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 App0 lab application is running.

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

S 41 W 01 Sr 41 R N P

ROM Bootloader has version 0x10.

6 Bootloader Protocol

The bootloader/monitor supports a binary protocol to:

- Download a program to the internal RAM.
- Perform ROM/RAM remap and reset.

6.1 Command Overview

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

Figure 3:
Command Overview

Command	Value	Meaning
Reserved	0 .. 0xF	Reserved are used for Status reporting
RESET	0x10	Reset to be done
RAMREMAP_RESET	0x11	Remap RAM to Address 0 and Reset
ROMREMAP_RESET	0x12	Remap ROM to 0 and start Bootloader
DOWNLOAD_INIT	0x14	Initialize for RAM download
W_RAM	0x41	Write RAM Region (Plain = Not encoded into e.g. Intel Hex Records)
ADDR_RAM	0x43	Set the read/write RAM 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 information 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:

Figure 4:
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 given 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.
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 Figure 4) 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 the 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 this is: ((0x10 + 0x00) XOR 0xFF) = 0xEF

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

6.4 Command List

Any valid command must have a value that is within the allowed range 0x10..0xFF. Note that not all numbers are valid commands. The numbers 0x0 .. 0xF are reserved for status information and are forbidden for commands. See also Figure 4 for further information about status values.

6.4.1 RESET

This command performs a SW 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. For more details please see the ARM Cortex M0 specifications of ARM.

Note: Depending on the value of the remap bit in the CCU, the system will start with the bootloader again (if ROM is mapped to address 0) or with 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.

Figure 5:
Reset Command

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

Possible Responses

- There is no STATUS set by the bootloader for this command in case the command was correct. = As 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).

Command is performed immediately without any delay.

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

Figure 6:
RAMREMAP Command

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

Possible Responses

- There is no STATUS set by the bootloader for this command in case the command was correct. = As 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.

Figure 7:
ROMREMAP Command

Address	Value	Meaning
CMD_STAT	0x12	REMAP ROM to 0 and RESET
SIZE	0	No parameters

Address	Value	Meaning
CSUM	0xED	

Possible Responses

● There is no STATUS set by the bootloader for this command in case the command was correct.
  = As 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 DOWNLOAD_INIT

This command is used to initialize the download HW for secure devices.

Figure 8:
ROMREMAP Command

Address	Value	Meaning
CMD_STAT	0x14	Initialize the HW for download to RAM
SIZE	1	
DATA0	0x29	Seed
CSUM	0xC1	

Possible Responses

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

6.4.5 W_RAM

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

Figure 9:
W_RAM Command

Address	Value	Meaning
CMD_STAT	0x41	Write to main 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.

Possible Responses

- 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.
- The command was incorrect an error is flagged in CMD_STAT.

6.4.6 ADDR_RAM

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

Figure 10:
W_RAM Command

Address	Value	Meaning
CMD_STAT	0x43	Specify the address of the next RAM read or 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.

Possible Responses

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

7   RAM Patch Download

The Bootloader application allows to download an encrypted image and start the downloaded image.

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

1.   Sent the DOWNLOAD_INIT command: S 41 W 08 14 01 29 C1 P.

2.   Set up the RAM address pointer with command ADDR_RAM.

3.   Send consecutive data with the command W_RAM.

4.   If you need to move the address pointer use command ADDR_RAM.

5.   Send consecutive data with the command W_RAM.

6.   Send the command RAMREMAP_RESET to start the downloaded application.

Simple example to download this Intel Hex snippet:

```
:020000042000DA
:100000006DC941853D15AA51F4D29EA8A7AC77E9E8
:10001000F9EC202463B8F1A50BA765B432B818D762
…
:101C1000FF8000D6EAF77C36807C00FF5D488E5D51
:0400000520000069 6E
:00000001FF
```

As I²C Strings:

Send DOWNLOAD_INIT

S 41 W 08 14 01 29 C1 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_0000, only lower 16 bits are used of address (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

As soon as the App0 is running the following I²C strings can be used for communication with the application:

1. Read out App0 Version, HW Version and the serial number of the HW

2. Configure App0

3. Start App0

4. Read back results

5. Stop App0

6. Trim Oscillator

8.1 Factory Calibration

The device has to 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
● Clear glass (no smudge on the glass)
● No target in front of the device within 40 cm (see datasheet)
● Dark room or low ambient light

The factory calibration sequence is started by issuing to the following command to the TMF8X0X:

S 41 W 10 0A P

After this factory calibration is started. You can wait for the 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

or have to wait a maximum time of 2 seconds.

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 A N P

E.g. my device reads back this data:

S 41 W 20 Sr 41 R 01 17 00 ff 04 20 40 80 00 01 02 04 00 fc P

This information has to be written to the TMF8X0X every time after a power cycle as configuration – see also section 8.3.

8.2 App0 Version

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 than 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

8.3 Configure the App0

This example downloads the calibration and state data to the App0.

Calibration Data example (you have to use the data you stored from the factory calibration – see 8.1):

S 41 W 20 01 17 00 ff 04 20 40 80 00 01 02 04 00 fc P

State Data example:

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

8.4 Additional Configuration for App0 / Interrupt Suppression

Information

This feature is only available for TMF880x devices.

There are 2 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. In 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. 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 = unsigned short (0..65535)
- High-threshold = unsigned short (0..65535)

Note: If the low-threshold is set to a bigger value than the high-threshold, no object can be reported.

Command to write 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.

Figure 11:
Command WR_ADD_CONFIG

Name	I²C Register	Description
cmd_data4	0x0b	Persistence value
cmd_data3	0x0c	Least significant byte of the low-threshold
cmd_data2	0x0d	Most significant byte of the low-threshold
cmd_data1	0x0e	Least significant byte of the high-threshold
cmd_data0	0x0f	Most significant byte of the high threshold
Command	0x10	WR_ADD_CONFIG command must be written here (0x08)

E.g. for a persistence of 5, low_threshold of 55 mm and high_threshold of 500 mm (55 = 0x37, 500 = 0x1F4)

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

The device will write back to register 0x10 a 0x00 and to register 0x11 a 0x08 to confirm that the command was accepted.

Command to read thresholds RD_ADD_CONFIG = 0x09:

The command has no parameters.

After the execution of the command (i.e. register 0x11 has to read back as 0x09), the host can read out the parameter from the registers:

Figure 12:
Response to Command RD_ADD_CONFIG

Name	I²C Register	Description
REGISTER_CONTENTS	0x1e	Reading 0x09 means Additional configuration is given in regsiters 0x20..0x24
TID	0x1f	Transaction ID

Name	I²C Register	Description
PERSISTENCE	0x20	Persistence value
LOW_THRESHOLD_LSB	0x21	Least significant byte of the low threshold
LOW_THRESHOLD_MSB	0x22	Most significant byte of the low threshold
HIGH_THRESHOLD_LSB	0x23	Least significant byte of the high threshold
HIGH_THRESHOLD_MSB	0x24	Most significant byte of the high threshold

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 reads back as 09

To 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

8.5 Start the App0

8.5.1 TMF8701

Use above configuration and configure for continuous mode, period of 100 ms, GPIOs are not used, run combined proximity and distance algorithm.

S 41 W 08 03 23 00 00 00 64 ff ff 02 P

Explanation (see datasheet for details):

cmd_data7 = 03	Algorithm state and factory calibration is provided
cmd_data6 = 23	Run proximity and distance algorithm and combine histograms for distance
cmd_data5 = 00	No GPIO control used
cmd_data4 = 00	No GPIO control used
cmd_data3 = 00	Needs to be always 00
cmd_data2 = 64	Repetition period in ms 64 hex = 100 ms
cmd_data1 = ff	Needs to be always ff
cmd_data0 = ff	Needs to be always ff
command executed = 02	Execute distance and proximity detection

8.5.2 TMF8801 and TMF8805

Use above configuration and configure for continuous mode, period of 100 ms, GPIOs are not used

S 41 W 08 03 23 00 00 00 64 D8 04 02 P

Explanation (see datasheet for details):

cmd_data7 = 03	Algorithm state and factory calibration is provided
cmd_data6 = 23	Combine the capture of the short and long distance histograms
cmd_data5 = 00	No GPIO control used
cmd_data4 = 00	No GPIO control used
cmd_data3 = 00	Object detection threshold, use 00 as default
cmd_data2 = 64	Repetition period in ms 64 hex = 100 ms
cmd_data1 = D8	Number of iterations, low byte; 1 LSB = 1 k
cmd_data0 = 04	Number of iterations, high byte; 1 LSB = 1 k*256 Set to 1.2 M iterations
command executed = 02	Execute distance and proximity detection

8.6 Read Results

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

Register 0x1d report the status of TMF8X0X; the detailed explanation is shown in section 12.1.

If the content of register 0x1E is 0x55 the registers 0x20 and following results contain the result.

The host should check that the number in register 0x20 counts up every time a new result is calculated. See also the TMF8X0X datasheets for interpretation of the registers.

A new result will be reported by the TMF8X0X every 100 ms in this example. Please note that the host has to compensate for oscillator drift between host and TMF8X0X.

> **! Attention**
>
> Always start reading from 0x1D onward to correctly report sys_clock. If state data needs to be readout, simply continue reading. See datasheets for details

8.7 Stop the App0

S 41 W 10 ff P

After issuing the command, the application will terminate as quick as possible. This time depends on the internal state.

8.8 Readout of Raw Histograms

In order to readout the raw histogram, the TMF8X0X needs to be configured in this mode. The procedure is as follows:

Stop any command:

S 41 W 10 FF P

Read back previous command, should return 0xff.

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 TMF8801 or TMF8805 – see datasheet for available histograms).

S 41 W 0C 10 00 00 00 30 P

Configure for cyclic measurement - no calibration data no state data.

S 41 W 08 00 23 00 00 00 80 D0 07 02 P

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 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 I^{2}C blockread (128 bytes). Otherwise, the result can be corrupted.

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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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 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).

9   Timings

9.1   Startup and Bootloader Timings

All timings in the table below are given starting by 0 when the Enable line is pulled high. If two lines show the same timestamp, there is no wait time required between these 2 lines. The execution should follow the order given in the table.

Figure 13:
Timings

Time	I²C String	Description
0 ms	Enable Power Line (set HIGH)	Power ToF, HW starts, Bootloader puts ToF into Power Down mode
1.5 ms	Do not send any I²C commands	I²C slave is available (Registers 0xE0..0xFF only)
3 ms	S 41 W E0 01 P	PON=1 = Wake-up ToF
5 ms	S 41 W E0 Sr 41 R N P	CPU is ready (read register 0xE0 as 0x41)
5 ms	S 41 W 00 Sr 41 R A A A N P	Read out the Bootloader APP ID + Version (80 10 80 00)
5 ms	S 41 W 08 Sr 41 R A A N P	Read out status of bootloader (00 00 FF)
5 ms	S 41 W 08 14 01 29 C1 P	Send Download Init command
5.15 ms	S 41 W 08 Sr 41 R A A N P	Read out status of bootloader (00 00 FF)
5.15 ms	S 41 W 08 43 02 00 00 BA P	Send ADDR_RAM command for address 0
5.30 ms	S 41 W 08 Sr 41 R A A N P	Read out status of bootloader (00 00 FF)
5.30 ms	S 41 W 08 41 10 4D B9 42 …66 P	Send W_RAM command with 16 bytes payload.
5.45 ms	S 41 W 08 Sr 41 R A A N P	Read out status of bootloader (00 00 FF)
		Repeat above 4 lines until end of download is reached. You can only repeat the last 2 steps if you do have a continous data stream.
X.00 ms	S 41 W 08 Sr 41 R A A N P	Read out status of bootloader (00 00 FF)
X.00 ms	S 41 W 08 11 00 EE P	Send RAMREMAP_RESET command
X.02 ms	S 41 W E0 Sr 41 R N P	Read back register 0xE0 shows 0x01 (CPU Busy)
+1 ms	S 41 W E0 Sr 41 R N P	Read back register 0xE0 shows 0x41 (CPU ready), APP_0 is read to talk to host

> **Information**
>
> If you send a command W_RAM with 16 bytes you have to 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 have to wait for 1 millisecond for the status to become ready.

9.2 APP0 Timings

HW Version is available immediately, as well as the SW Version.

Read out of the serial number is SW assisted, so the wait time after issuing the I²C string S 41 W 10 47 P is 500 microseconds. After this time the serial number is available.

Configuration of the application requires no wait timing.

After start of the application, an interrupt is triggered after the configured period in milliseconds. However the oscillator drift has to be taken into consideration so the interrupt can occur ± 4% later/earlier.

The result data is available when the interrupt occurs.

After issuing the Stop command, the application will terminate as quick as possible. This time depends on the internal state. In the worst case a break-down voltage detection is occurring and this cannot be interrupted and may take up to 8 milliseconds.

10   Oscillator Drift Correction

The firmware for TMF8X0X was developed for an oscillator frequency of 5 MHz.

As the internal RC oscillator has a production variation, the reported distance should be corrected with this formula:

real_distance = reported_distance * Relation_Host_to_TMF8X0X

where for any time interval, the amount of ticks that have expired at the host are collected as Host_ticks, and the amount of ticks that have expired at the TMF8X0X in the same interval are collected as TMF8X0X_ticks:

Relation_Host_to_TMF8X0X = Host_ticks / TMF8X0X_ticks

The TMF8X0X reports its internal timestamp since it has been activated (App0 and PON=1). The resolution of the timestamp is 0.2 microseconds. The reported value is an unsigned 32-bit value that wraps around approximately every 14 minutes.

The timestamp is reported by the TMF8X0X in registers SYS_CLOCK_0, SYS_CLOCK_1, SYS_CLOCK_2, SYS_CLOCK_3 in little endian format. To get this information you have to perform a block read starting from address 0x1D (see also the Register Description sections of the TMF8X0X datasheets).

Figure 14 shows a typical example for read out of the timestamps and putting them in relation to the used host in this example:

Figure 14:
Typical Timing Correction Example

TMF8701 TS	Host TS	TMF8701	Host TS	TMF8701	Host	Relation
[0.2 uSec]	[16 uSec]	[100uSec]	[100uSec]	TS[n] - TS[n-4]	TS[n] - TS[n-4]	Host/TMF8701
3004720	9707909	6009	1553265			
3876069	9718034	7752	1554885			
4774353	9728472	9548	1556555			
5656482	9738722	11312	1558195			
6549396	9749097	13098	1559855	7089	6590	0.929609
7447601	9759535	14895	1561525			
8329690	9769785	16659	1563165			
9201051	9779910	18402	1564785			
10093901	9790285	20187	1566445			
10981477	9800598	21962	1568095	7067	6570	0.929673
11858390	9810786	23716	1569725			
12756772	9821223	25513	1571395			
13655122	9831661	27310	1573065			
14553555	9842098	29107	1574735			
15494970	9853036	30989	1576485	7273	6760	0.929465
16409603	9863661	32819	1578185			
17302631	9874036	34605	1579845			
18195681	9884411	36391	1581505			
19088594	9894786	38177	1583165			
19987011	9905224	39974	1584835	7155	6650	0.929420
20863797	9915412	41727	1586465			
21832014	9926661	43664	1588265			
22719523	9936974	45439	1589915			
23617677	9947412	47235	1591585			
24505041	9957724	49010	1593235	7283	6770	0.929562
25403214	9968162	50806	1594905			
26295994	9978537	52591	1596565			
27264032	9989787	54528	1598365			
28205336	10000725	56410	1600116			
29103608	10011163	58207	1601786	7401	6881	0.929739
29996466	10021537	59992	1603445			
30878548	10031788	61757	1605086			
31857381	10043163	63714	1606906			
32755597	10053600	65511	1608576			
33696870	10064538	67393	1610326	7401	6881	0.929739
34659672	10075725	69319	1612116			
35541881	10085976	71083	1613756			
36445676	10096476	72891	1615436			
37344128	10106914	74688	1617106			
38242473	10117351	76484	1618776	7165	6660	0.929518
39135411	10127726	78270	1620436			
40092870	10138851	80185	1622216			

The host (in this example is running with 16 MHz) reports its own clock frequency as time stamps with a granularity of 16 microseconds per tick. The host reads out the timestamps from TMF8X0X and host periodically. In this example roughly every 160 milliseconds.

The timestamp relation is calculated as follows:

$(Host\_ts[n] – Host\_ts[n-4]) / (Tmf8X0X\_ts[n] – Tmf8X0X\_ts[n-4])$

I.e. the difference between 5 samples is used for the relation calculation.

The accuracy depends on the jitter between I²C communication happening and the host time stamping.

11 Oscillator Re-Trimming

In case that oscillator re-trimming is required, the following procedure has to 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 to register 0xE0 the value 0x00).

4. Wait for the device to enter standby mode (read register 0xE0 until it reads backas 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 to register 0xE0 the value 0x01).

10. After this procedure the device will operate with the changed frequency.

! Attention

Do not change any other bit than bit 6 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.

---

i Information

Re-trimming the oscillator outside the allowed range of 4.5 MHz – 5.5 MHz can lead to a lockup of the device.

It is recommended to perform small steps (range: ±1... ±8).

---

i Information

Running the TMF8801/05 above 4.75 MHz 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.

Note 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 15:
Code Example for Oscillator Re-Trimming

```c
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 16:
Nominal Oscillator Trim Values

Plot 37: oscillator frequency range & availability
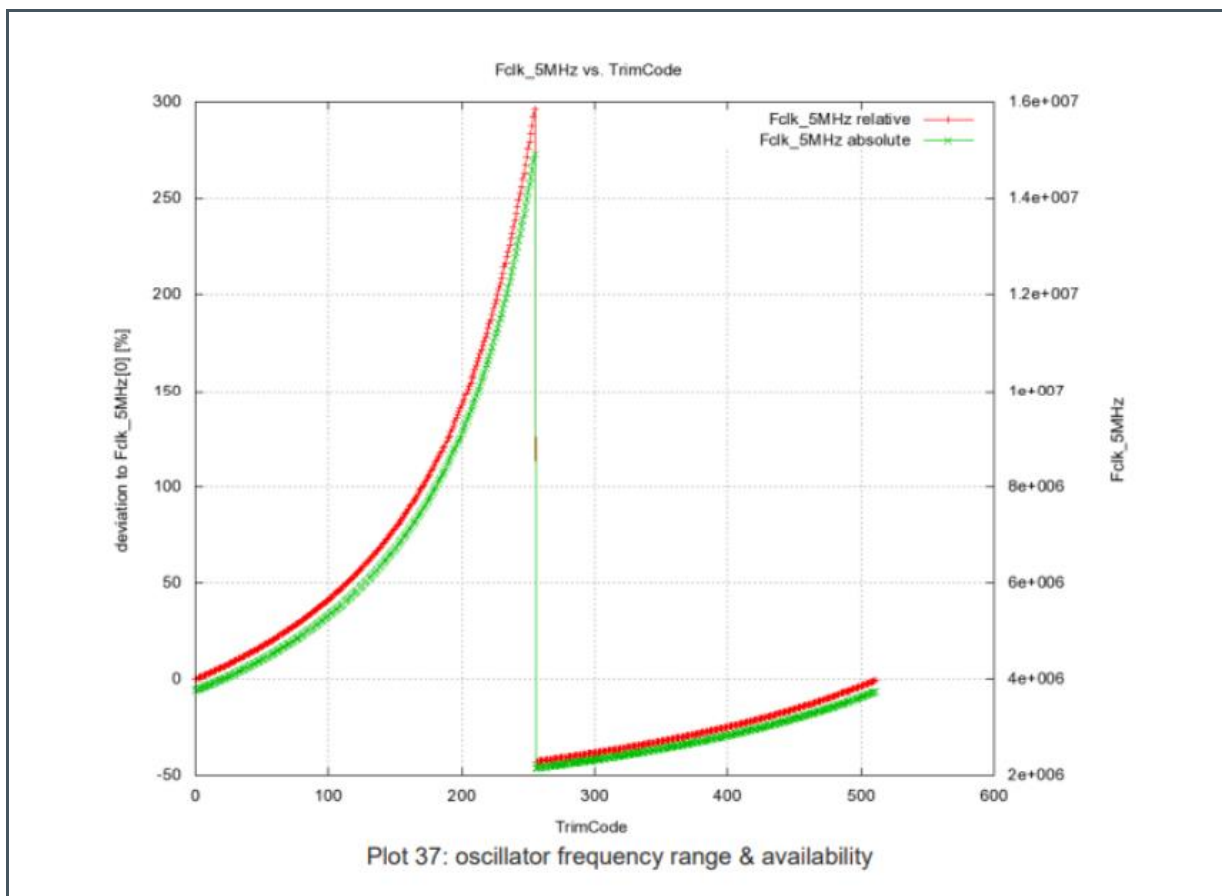
Note that the trim value is a signed 9-bit value, the picture above shows it as an unsigned 9-bit value. The unsigned 9-bit values [256 .. 511] correspond to [-1 .. -256] 9-bit signed values.

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 5 MHz.

The device is trimmed correctly during production test.

Adjusting the trim-value always has to 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.

There are two methods available to do this:

1. By using individual enable lines to each device.

2. By controlling GPIO0 and/or GPIO1 lines to each device.

For (1.) you need one individually controllable GPIO line for each ToF device on the same I²C bus.

For (2.) you need at least one individually controllable GPIO line to the first ToF device on the same I²C bus. The other devices on the same I²C bus must then be daisy chained through their GPIO pins.

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 to drive each GPIO line high or low.

Figure 17:
Enable Line Control Schematic for Dynamic I²C Address Change
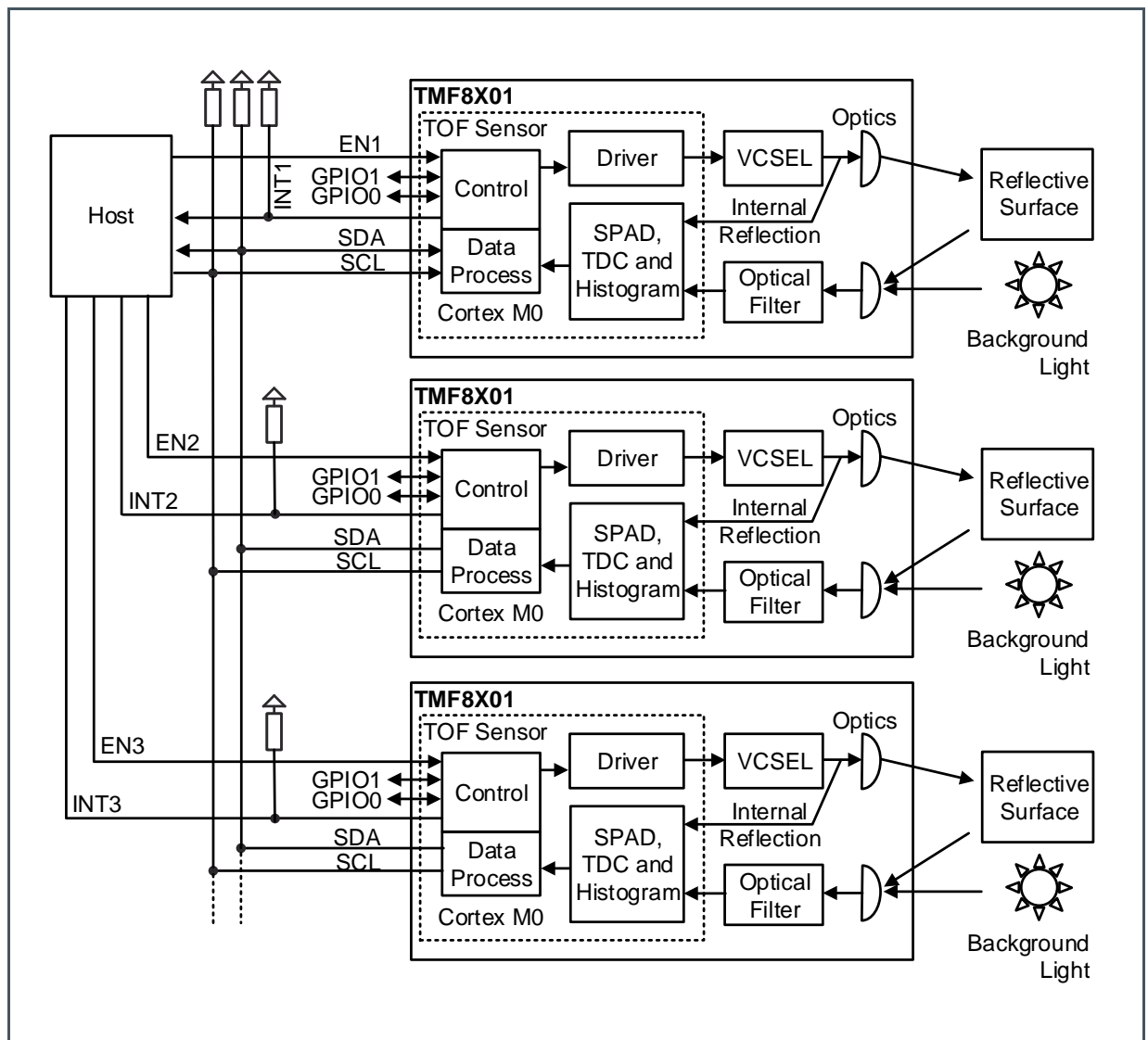
In below example, there are four ToF devices that are being reprogrammed to use different addresses. The host driver drives all enable lines low. All ToF devices are in Power Off state.

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 sends the bootloader a DOWNLOAD_INIT command.

5.   The host driver downloads the patch through I²C bootloader commands.

6. The host driver tells the bootloader to perform a RAMREMAP_RESET command.

Now the ToF device is ready to be reprogrammed.

7. 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

8. 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

9. 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.

12.1.2 Cascaded GPIO Line Controlled

This section describes how to perform the address change when having all devices connected to a shared enable line, and daisy chaining the ToF devices through GPIO lines.

Figure 18:
Cascaded GPIO Line Control for Dynamic I²C Address Change
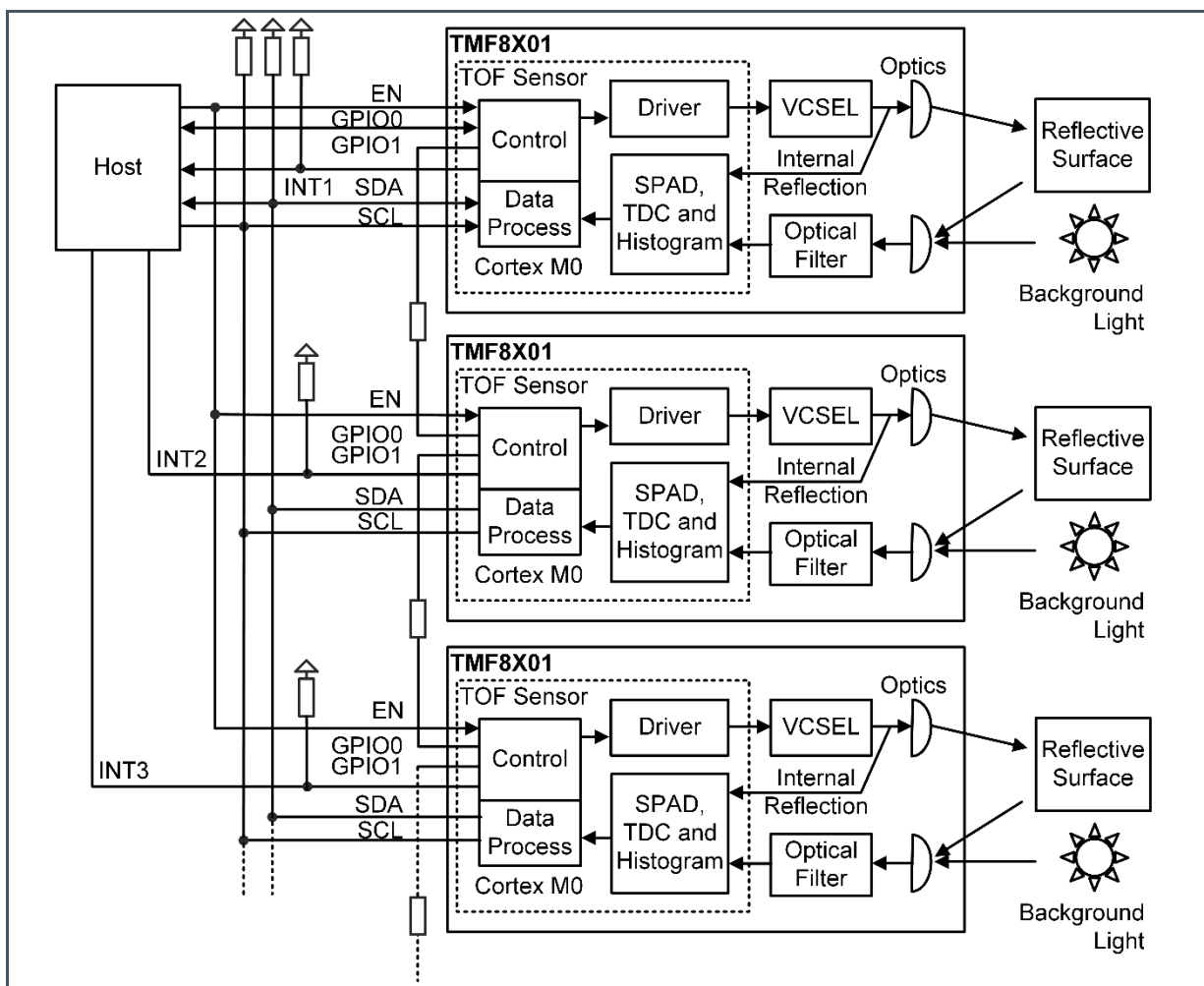
ⓘ Information

It depends on the host driver if individual interrupt lines are necessary. If not, a single combined interrupt line is sufficient. If the host driver uses polling, remove the interrupt line.

Below example uses four ToF devices and programs them to different addresses. The host driver has a GPIO line connected to the GPIO0 of device #1. The 4 ToF devices are daisy chained by connecting GPIO1 of device #1 to GPIO0 of device #2, GPIO1 of device #2 connected to GPIO0 of device #3, GPIO1 of device #3 connected to GPIO0 of device #4.

The host driver drives the enable line low. All ToF devices are in Power off state.

1. The host driver drives its own GPIO line (which is connected to GPIO0 of the first ToF device) LOW.

2. The host driver drives the enable line high. All ToF devices will enter the power down state.

3. The host driver sends the following I²C string:
S 41 W E0 01 P

4. The host driver reads back until the I²C register 0xE0 changes to the value 0x41

5. The host driver sends the bootloader a DOWNLOAD_INIT command.

6. The host driver downloads the patch through I²C bootloader commands.

7. The host driver tells the bootloader to perform a RAMREMAP_RESET command.

Now all ToF devices are ready to be reprogrammed.

8. Program all 4 devices to have on GPIO0 as input and GPIO1 as output LOW. Send this I²C string:
S 41 W 0f 40 0f P

9. Program all 4 devices to wait for a HIGH on GPIO0 and a do not care on GPIO1. And if a device sees that pattern it shall reprogram itself to use I²C slave address 0x51:
S 41 W 0e a2 05 49 P

10. Now the host drives its GPIO line HIGH

11. Now the host driver must send a stop command to make sure the ToF devices wake up and check the GPIO lines:
S 41 W 10 ff P

12. Now you can check that a device is listening to the new address.
S 51 W e0 Sr 51 R A P

13. Now the host driver should drive GPIO line LOW again.

14. Program all remaining 3 devices to wait for a HIGH on GPIO0 and a do not care on GPIO1. And if a device sees that pattern it shall reprogram itself to use I²C slave address 0x52:
S 41 W 0e a4 05 49 P

15. Now tell the device that has just been reprogrammed to 0x51 to drive its GPIO1 line HIGH:
S 51 W 0f 50 0f P

16. Now the host driver must send a stop command to make sure the ToF devices that are still listening to default slave address wake up and check the GPIO lines:
S 41 W 10 ff P

17. Now you can check that a device is listening to the new address.
S 52 W e0 Sr 52 R A P

18. Now the host driver should tell device 0x51 to drive GPIO1 line LOW again.
S 51 W 0f 50 0f P

19. Program all remaining 2 devices to wait for a HIGH on GPIO0 and a do not care on GPIO1. And if a device sees that pattern it shall reprogram itself to use I²C slave address 0x53:
S 41 W 0e a6 05 49 P

20. Now tell the device that has just been reprogrammed to 0x52 to drive its GPIO1 line HIGH:
S 52 W 0f 50 0f P

21. Now the host driver must send a stop command to make sure the ToF devices that are still listening to default slave address wake up and check the GPIO lines:
S 41 W 10 ff P

22. Now you can check that a device is listening to the new address.
S 53 W e0 Sr 53 R A P

23. Now the host driver should tell device 0x52 to drive GPIO1 line LOW again.
S 52 W 0f 50 0f P.

24. Program the one remaining device to wait for a HIGH on GPIO0 and a do not care on GPIO1. And if a device sees that pattern it shall reprogram itself to use I²C slave address 0x54:
S 41 W 0e a8 05 49 P

25. Now tell the device that has just been reprogrammed to 0x53 to drive its GPIO1 line HIGH:
S 53 W 0f 50 0f P

26. Now the host driver must send a stop command to make sure the ToF devices still listening to default I²C slave address wake up and check the GPIO lines:
S 41 W 10 ff P

27. Now you can check that a device is listening to the new address.
S 54 W e0 Sr 54 R A P

28. Now the host driver should tell device 0x53 to drive GPIO1 line LOW again.
S 53 W 0f 50 0f P

12.2 Status Register Address 0x1D

Register 0x1D gives the internal status of the application running on TMF8X0X. 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.

TMF8X0X update the status register every time the internal state machine traverses through states, only errors are sticky and remain.

Figure 19:
Status Register Codes

Status	Value	Description
Idle	0x00	No error, information that internal state machine is idling.

Status	Value	Description
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.
BdvGenError	0x13	Deprecated, incorrect breakdown voltage mode selected. SW uses BDVSafe mode now.
UnusedCmd	0x14	Deprecated, no longer in use.
BdvErrorLow	0x15	Deprecated, BDV below lower limit. SW uses BDV Safe mode now, which always returns a BDV.
BdvErrorHigh	0x16	Deprecated, BDV above upper limit. SW uses BDV Safe mode now, which always returns a BDV.
BdvComperatorErr	0x17	Deprecated, only BDV windowing algorithm which is no longer used could report this error.
(HistRam) Invalid Parameter Error	0x18	Error, internal program error. A parameter to a function call was out of range.
(InvalDevice) HAL Interrupted	0x19	A status information that a measurement got interrupted. Is only used internally. Not populated to I²C register.
InvalParam	0x1a	Deprecated, HistRam (value 0x18) is used instead.
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.
InvalState	0x1d	Error, internal program error. The constant state table is faulty.
Unknown	0x1e	Deprecated, was only used in special test build.
ErrAlgorithm	0x1f	Internal error in algorithm.
ErrDupAlloc	0x20	Deprecated, was used in previous SW version only.
ErrNoMem	0x21	Memory allocation routine failed. Should not occur as deterministic memory allocation is used.
ErrMemIdNotFound	0x22	Deprecated, was used in previous SW version only.

Status	Value	Description
InvalData	0x23	Only used in test modes.
HalInterrupted	0x24	Same as 0x19
MissingCallback	0x25	Unused
ErrFactCalib	0x26	Deprecated
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.
ErrInvalidAlgState	0x29	Parsing of the provided algorithm state found an illegal algorithm state value.
ErrInvalidProxConfig	0x2a	Only used in test modes.
ErrInvalidDistConfig	0x2b	Only used in test modes.

13 Revision Information

Changes from previous version to current revision v8-00	Page
Added additional configuration section for interrupt suppression feature	22-23

- 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.

14   Legal Information