Application Note

AN001015

TMF8820/TMF8821

Host Driver Communication

v3-00 • 2021-Aug-31

Content Guide

1 Introduction

The purpose of this document is to describe the communication between host driver and the TMF8820 and TMF8821. Most of the communication is the same for TMF8820 and TMF8821. Whenever there is a difference, the device is named either TMF8820 or TMF8821. To address both devices the name TMF882X will be used in this document.

1.1 I²C Nomenclature

The primary interface of the TMF882X is I²C. Please refer to the TMF8820/TMF8821 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.

The default I²C 7-bit slave address (un-shifted) is 0x41. This document will use the following nomenclature for description of the communication:

- 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.

If binary values are used in this document, the following format will be used:

b_0101_111x

Where ‘x’ is a “do not care”, the left-most bit is bit 7, the right-most bit is bit 0. The prefix ‘b’ is to give a hint that this is a binary number. The underscores are just for better readability.

1.2 General Information for Communication

This document describes the communication with the TMF882X ROM version 2. If your device has another ROM version please contact **ams** for the correct document for your ROM version.

The primary communication interface for the TMF882X is I^2C, however the host may also have connected the INT pin to receive interrupts from the TMF882X. The TMF882X will de-assert the INT pin for all interrupt types it has been configured for in the register INT_ENAB (see section 4).

The TMF882X also has two general purpose IO pins that can be used for various tasks (see the datasheet and also section 4 which gives an example to configure a GPIO pin).

1.2.1 Power Up

The list below states the communication flow from power up:

1. Power the TMF882X
2. Pull the Enable Line HIGH
3. Write 0x01 to register 0xE0 = ENABLE
4. Poll register ENABLE until the value 0x41 is read back. See also section 2.1.
5. Read the register 0x00 = APPID to find out which application is running. See also section 2.4.
6. If the bootloader is running: APPID has value 0x80 then continue reading at section 3.
7. If the application is running: APPID has value 0x03 then continue reading at section 4.

The following chapters give more details about the different steps and the I^2C strings to be sent/received.

1.2.2 Enter STANDBY

To enter standby the host has to do the following steps:

1. If the measurement application is performing measurements, first stop the measurement application (see section 4.5).
2. Read back register ENABLE (0xE0) and check that the device is ready (reads back as b_01xx_0001).
3. Write to register ENABLE the value b_00xx_0000.
4. Read back register ENABLE until it has the value b_00xx_0010.

1.2.3 Wake Up Device (exit STANDBY)

To exit standby the host has to do the following steps:

1. Read back register ENABLE (0xE0) and check that the device is in STANDBY state (reads back as b_00xx_0010).
2. Write to register ENABLE the value b_00xx_0001.
3. Read back register ENABLE until it has the value b_01xx_0001.

1.2.4 STANDBY_TIMED

ROM v1 devices can only enter STANDBY state. The STANDBY_TIMED state is properly supported by ROM v2 devices only.

STANDBY_TIMED can only be achieved through the configuration of the measurement application. See the datasheet for details about the low-power option of register POWER_CFG (0x33) of the common configuration page.

Please refer to section 4.6 how to exit STANDBY_TIMED early.

2 Start Up

This chapter describes the startup behavior of the device. When the device is powered up and the ENABLE line is HIGH the device will see this as a cold-start.

Only after a power cycle or after an ENABLE line LOW for minimum of 1ms followed by ENABLE line HIGH the device will see it again as a cold-start.

All other resets will be treated as a warm start.

Independent of a cold start or a warm start the host should always make sure that the device is ready for communication and after that query which application the host is talking to.

2.1 Check That the Device Is Ready for Communication

The host should poll the register ENABLE (0xE0) to find out the state of the device itself, and if necessary wake-up the device through writing to register ENABLE.

When the host writes to register ENABLE, the host should always set the bits 5 and 4 to the same values it has read from the device.

2.1.1 Device is Ready if ENABLE Reads Back with Bit 6 Set

The device is only ready for host communication if bit 6 of register ENABLE is set.

I.e. a read out of register ENABLE

S 41 W E0 Sr 41 R N P

Should return the value: b_01xx_0001 to indicate that host access to all registers (not only register ENABLE) is possible.

2.1.2 If the ENABLE Register Reads Back with Bit 6 Cleared

If the bit 6 in register ENABLE is cleared, the device is not ready for host communication. The host must perform one of the following steps to enable communication.

ENABLE Reads Back as b_00xx_0001

If the value of ENABLE reads back as b_00xx_0001, then the host should continue to read the ENABLE register, as the CPU is currently in the process of initializing HW and SW. As soon as the CPU is ready the register value will change to b_01xx_0001.

ENABLE Reads Back as b_00xx_0010

If the value of ENABLE reads back as b_00xx_0010 the device is in STANDBY mode and requires writing of b_00xx_0001 to register ENABLE to wake up. The bits 5 and 4 should have the same value as you have read them from the device.

E.g. if the register ENABLE had the value: b_0010_0010 than the host should write the value b_0010_0001.

S 41 W E0 21 P

ENABLE Reads Back as b_00xx_0110

If the value of ENABLE reads back as b_00xx_0110 the device is in STANDBY_TIMED mode and will wake up due to the measurement timer expires (duration depends on the configuration of the measurement application), or the host can force the device to wake-up by writing b_00xx_0001 to the register ENABLE. However if the device has already woken up due to the measurement timer having expired in the meantime, this writing of the ENABLE register will be silently ignored and the measurement application will continue to perform measurements. The host in this case will have to send an explicit STOP command to abort the measurements and enter the IDLE state.

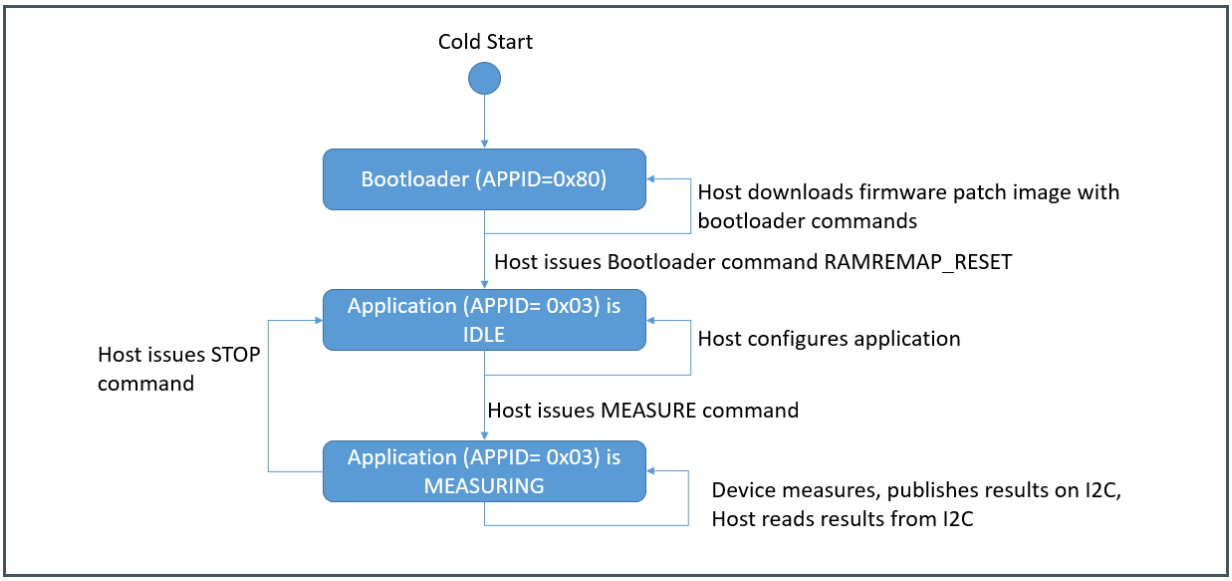
If the device was still asleep, this will be a forced wake-up and the measurement application will abort and enter the IDLE state.

2.2 Cold Start

When the device exits from a cold start the bootloader is always running.

Below is a simplified drawing of which applications are running after from cold start to the first measurement.

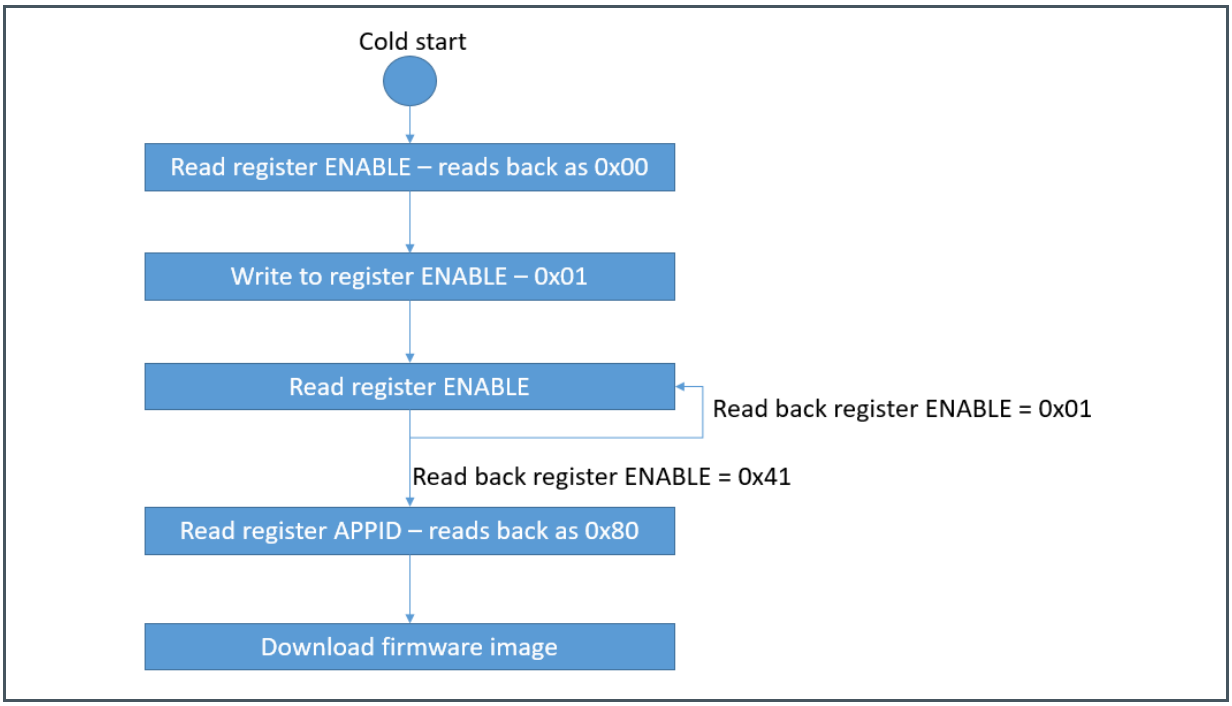
Figure 1:
Overview of Applications Running After a Cold Start



2.2.1 Communication After Cold Start

Below is a simplified drawing for the startup communication:

Figure 2:
Communication After a Cold Start



The host has to take the following steps to make sure it is talking to the bootloader:

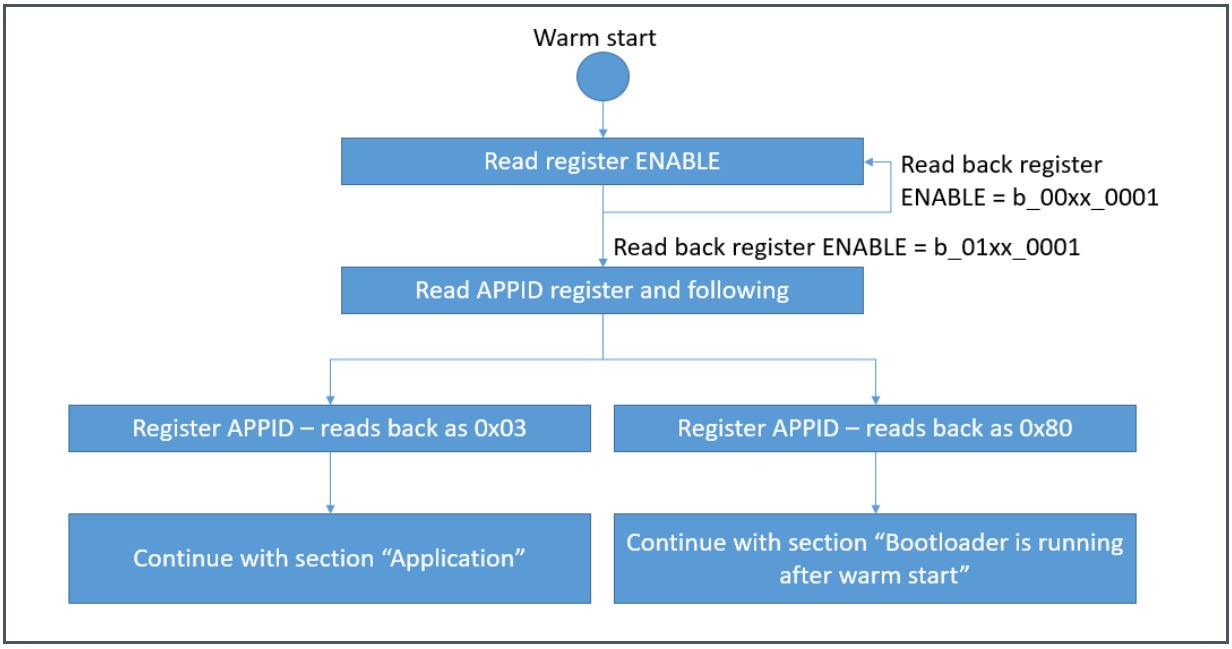
1. Check that the device is ready for communication (see section 2.1).
2. Read out the register APPID: S 41 W 00 Sr 41 R A N P
3. As after a cold-start the bootloader is running, the host should read back: 0x80 for APPID and version number should be 0x26 (for ROM v1 device) and 0x29 (for ROM v2 device).

The TMF882X ROM version 2 requires an firmware download before a valid measurement can be done (see section 3.2).

2.3 Warm Start

If the device exits with a warm-start normally the measurement application is running. Below is a simplified drawing for a warm-start:

Figure 3:
Communication After a Warm Start



The host should again check the APPID register to make sure that it is talking to the measurement application.

2.4 Check the Application ID

To check the application ID the host should read out the register APPID and also the version numbers to know if the correct firmware is running on the device.

S 41 W 00 Sr 41 R A A A N P

The application will return the following numbers:

Figure 4:
APPID and Version Numbers

Register Name	Value	Description
APPID	0x03	TMF882X measurement application
	0x80	TMF882X bootloader
MINOR	0x29	For the bootloader
	0x20	For the TMF8820 measurement application
	0x60	For the TMF8821 measurement application
PATCH	0x00	For the bootloader
	*	Depends on the firmware you have downloaded.

If the bootloader is running after a warm-start please read section 2.4.1 for further steps.

2.4.1 Bootloader is Running After a Warm Start

If the bootloader is running after a warm start the most likely reason is that the host has written to register ENABLE a value for bit 5 and 4 that differs from the read back value. The read back value after a successful firmware download from register ENABLE should have the following value:

b_0110_0001. I.e. bit 5 and 4 should have the binary value: b_10.

If the download was successful, but the host overwrote the bits 5 and 4 with a different value, the device can be switched back to measurement application mode. In this case, the host should perform the following sequence:

1. Put the device in STANDBY mode: S 41 W E0 20 P
2. Read back the register ENABLE until it reads back as 0x22: S 41 W E0 Sr 41 R N P
3. Write to the register ENABLE the value 0x21: S 41 W E0 21 P
4. Read back the register ENABLE until it reads back as 0x61: S 41 W E0 Sr 41 R N P

Now check that the application ID is 0x03 for TMF882X and that a proper communication with the measurement application is possible. E.g. load the common configuration page (see section 4).

If the application does not respond correctly, the device should be forced to a cold-start either through pulling the ENABLE line low or through a complete power cycle.

3 Bootloader

This section describes the communication with the bootloader application. The purpose of the bootloader is to load an image from the host to extend the functionality of the ROM code.

Make sure that the bootloader application is running on the device by reading the APPID register before downloading an image (see section 2.4).

The bootloader communication is protected by a simple checksum.

3.1 Checksum Calculation

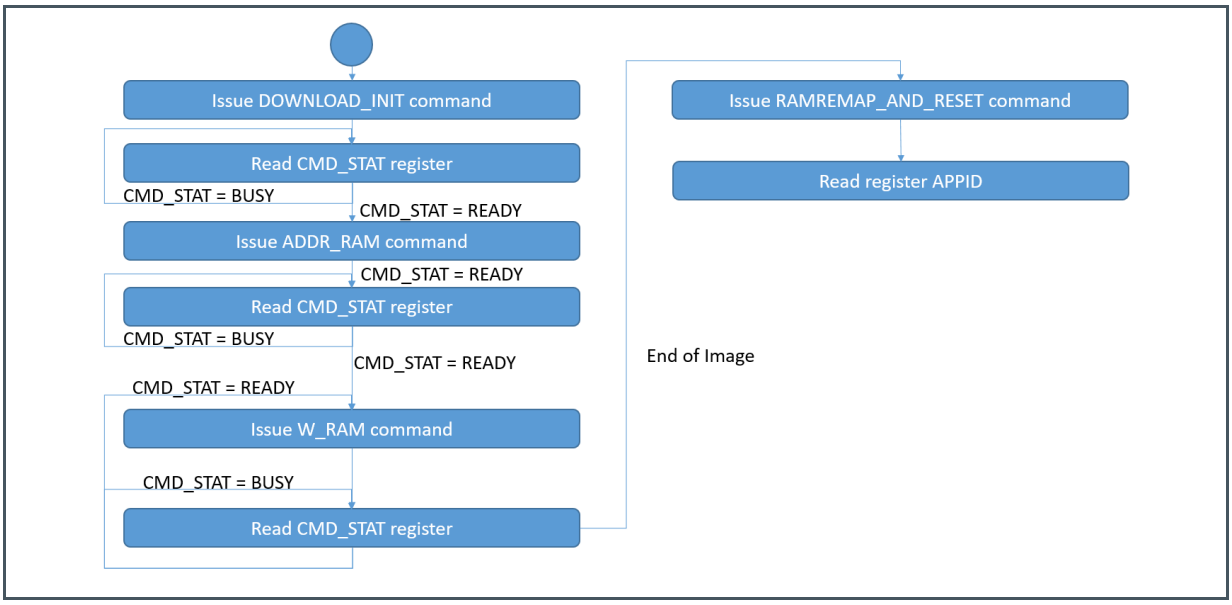
Each command and each response of the bootloader is protected by a simple checksum. The checksum is calculated as follows:
Calculate the sum of CMD_STAT + SIZE + (Sum of all Data-Bytes) and take the one's complement of the lowest byte of the sum. The one's complement was chosen so that the sum of zeros does not also get a checksum of zero.

E.g. for DOWNLOAD_INIT command it is: ((0x14 + 0x01 + 0x29) XOR 0xFF) = 0xC1

3.2 Image Download

Below is a simplified drawing for the image download:

Figure 5:
Image Download



The host has to initiate the device to accept an image. This is done by issuing the DOWNLOAD_INIT command.

S 41 W 08 14 01 29 C1 P

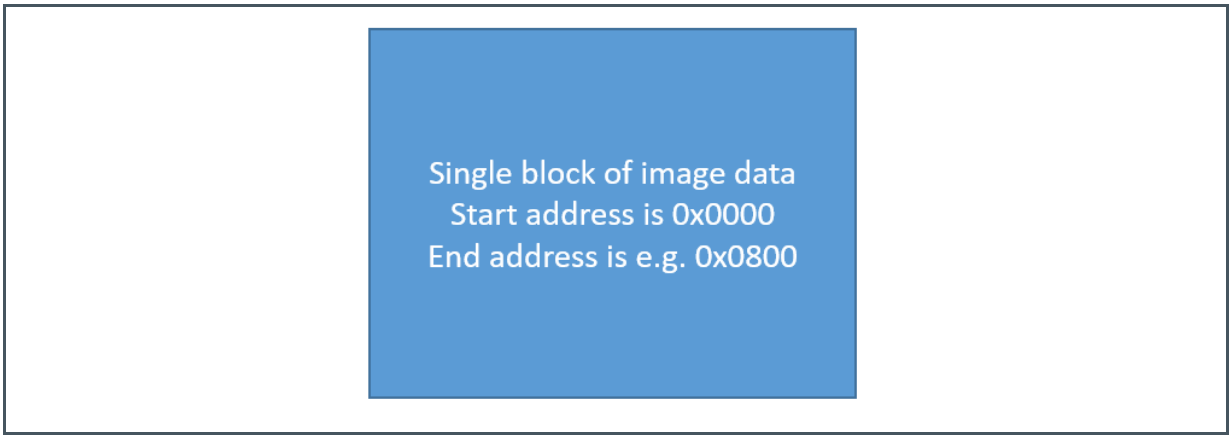
After this, the host should poll until the bootloader signals READY in register STATUS.

S 41 W 08 Sr 41 R A A N P

The host should wait until it reads back the following 3 bytes: 0x00 0x00 0xFF

The host has to set up the destination address for the image. If the image is one continuous block of data that will be downloaded, then this has to be done only for the first data block. I.e. your image will look like this:

Figure 6:
Single Block of Image to Load



Below is the string to set the destination address to 0x0000 with the command SET_ADDR:

S 41 W 08 43 02 00 00 BA P

After this, the host should poll until the bootloader signals READY in register CMD_STAT:

S 41 W 08 Sr 41 R A A N P

The host should wait until it reads back the following 3 bytes: 0x00 0x00 0xFF

Now the host can load the image with a series of the W_RAM commands. E.g. to load 20 bytes 0x7F, 0x7E, 0x7D, … 0x71, 0x70, 0x5F, 0x5E, …. 0x51, 0x50 to the RAM the following string has to be issued:

S 41 W 08 41 20 7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71 70 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51 50 AE P

After this, the host should poll until the bootloader signals READY in the CMD_STAT register:

S 41 W 08 Sr 41 R A A N P

The host should wait until it reads back the following 3 bytes: 0x00 0x00 0xFF

Now the host can send the next data packet to the device with the command W_RAM, and after that check again the CMD_STAT register.

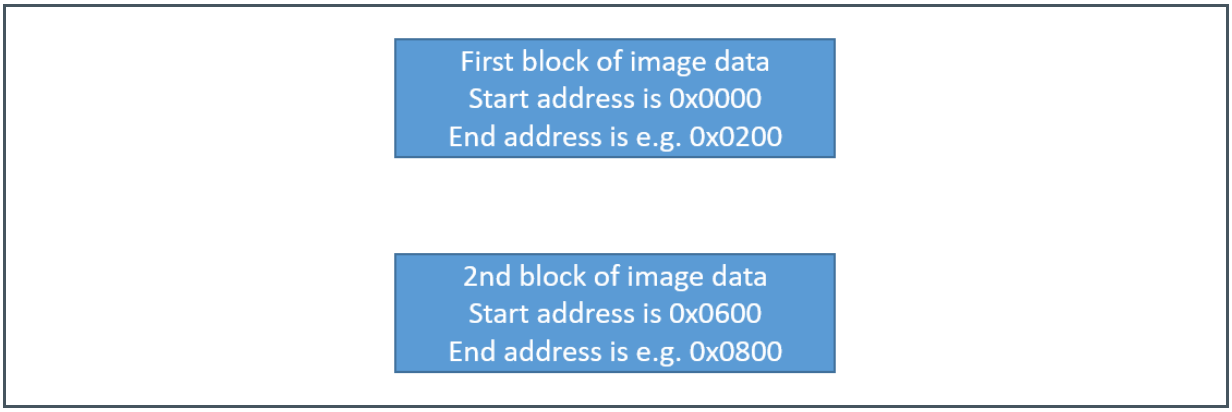
When all data packets have successfully been downloaded the host must issue a RAMREMAP_RESET command and wait for the APPID to become the measurement application ID.

S 41 W 08 11 00 EE P

If after a maximum wait time of 2.5 ms the register APPID does not read back with the measurement application ID, the download failed and the host should power cycle the device (see also section 3.2.1).

If the image is not a single continuous block the SET_ADDR command has to be issued for each block of continuous data. Such an image would look like this:

Figure 7:
Image Consisting of 2 Blocks of Data to Be Downloaded



The communication flow would be like in the list below:

1. Issue DOWNLOAD_INIT command
2. Wait for bootloader READY in CMD_STAT register
3. Issue SET_ADDR command for address 0x0000
4. Wait for bootloader READY in CMD_STAT register
5. Issue W_RAM command with the first chunk of data e.g. 128 bytes.
6. Wait for bootloader READY in CMD_STAT register
7. Issue W_RAM command with the next chunk of data e.g. 128 bytes.
8. Wait for bootloader READY in CMD_STAT register
9. Repeat steps 7. and 8. until the first 0x200 bytes are transmitted.
10. Issue SET_ADDR command for address 0x600
11. Wait for bootloader READY in CMD_STAT register
12. Issue W_RAM command with the first chunk of data from the 2nd data block e.g. 64 bytes
13. Wait for bootloader READY in CMD_STAT register.
14. Issue W_RAM command with the next chunk of data from the 2nd data block e.g. 64 bytes
15. Wait for bootloader READY in CMD_STAT register.
16. Repeat steps 14. and 15. until the complete 2nd data block is transmitted
17. Issue REMAPRAM_RESET command
18. Wait for the measure application ID to be read back from register APPID

3.2.1 If a Download Fails

There can be several reasons why an image download is not successful. Here is a list of reasons together with ways how to investigate the issue and solve it.

- The host did not check that the bootloader was ready to accept a new command after each command.

If you write to the bootloader CMD_STAT register before it is ready to accept a new command, the command was issued too fast and is silently discarded by the bootloader. I.e. one of your data records may have got lost and the image is incomplete and cannot execute correctly.

Solution: Make sure your host driver always checks for a command to be successfully completed.

- The checksum of a command is wrong.

If a command is sent without a checksum or with an incorrect checksum, the bootloader will not accept the command and flag an error in the CMD_STAT register.

Solution: Make sure you calculate the checksum for each command correctly. The checksum calculation is described in section 3.1.

- A bootloader error is ignored.

If the bootloader reports an error as status for a command in the register CMD_STAT, this should be treated seriously. The bootloader did not execute the command. If an error is simply ignored the likelihood of the image being correct is very low.

Solution: Make sure you abort the firmware download and investigate the reported error. The bootloader section of the datasheet will give details about the individual commands and corresponding errors. Fix the failing command and try the download again.

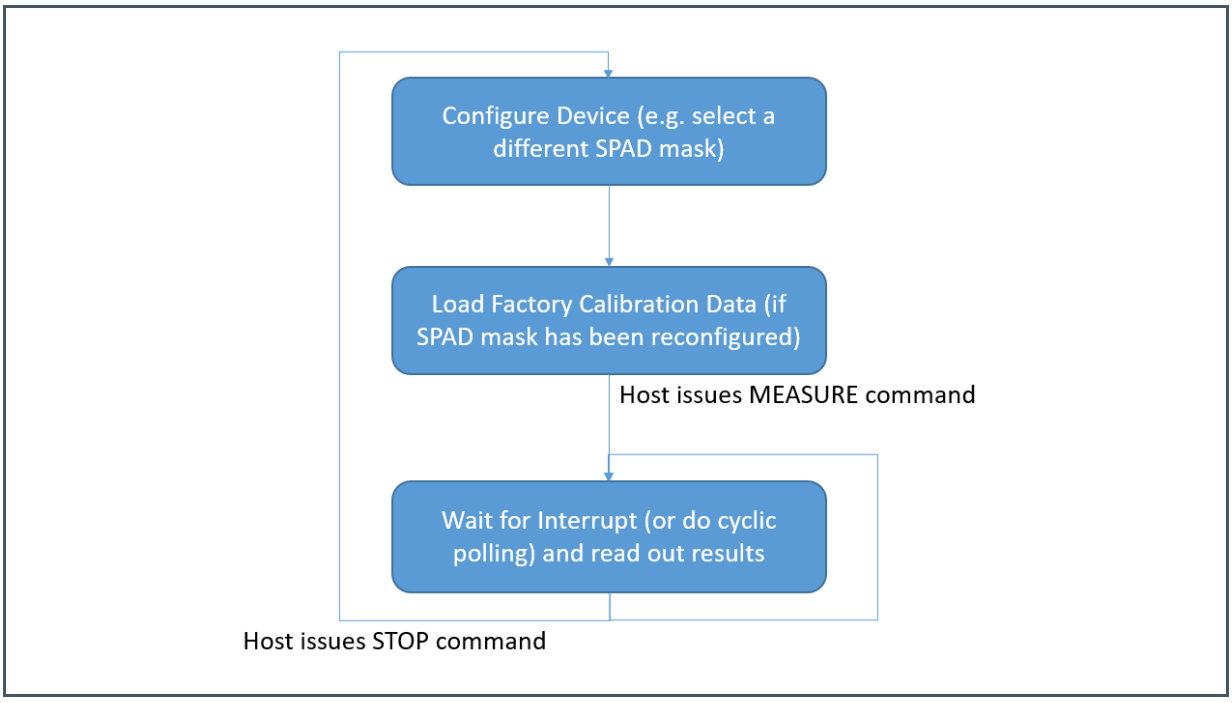
4 Application

This section describes the communication with the measurement application. To make sure that the measurement application is running the host should check the content of the register APPID (see also section 2.4).

A read back of 0x03 indicates that the measurement application is running on the TMF882X.

The picture below shows the general communication flow for the measurement application. The device can be configured (if the default values do not match the desired configuration). Depending on the selected SPAD mask, the correct (matching) factory calibration data needs to be loaded to the device. After this, the host can issue the MEASURE command and the application will produce distance results according to the configuration.

Figure 8:
Measurement Application Communication Overview



The TMF882X uses a command – status based protocol for communication with the host. The general layout of the I²C registers is like the following:

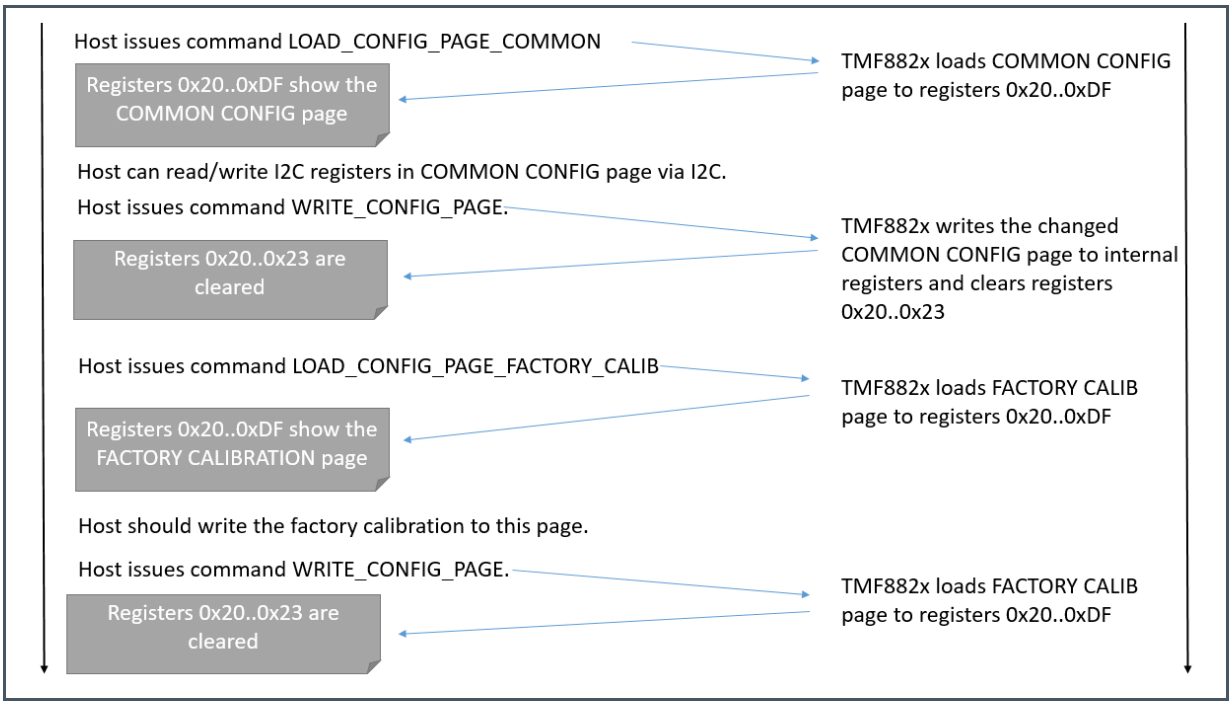
Figure 9:
Brief Overview of I²C Register Layout

Address(es)	Name	Description
0x00..0x03	APPID and VERSIONS	These 4 registers provide version information about the firmware that is running on the TMF882X.
0x04..0x07	*_STATUS	These 4 registers give status information of the major firmware blocks of the device. If the CMD_STAT register has an ERROR or WARNING value these 4 registers will give more details to the root cause of the error or warning. Please refer to the datasheet for full details about the values and their meaning.
0x08	CMD_STAT	The host writes commands to this register. A command has the value range of 0x10 .. 0xFF. The device will write the status back to this register. Status values have the range of 0..0xF. With 0=STAT_OK (meaning command successfully executed), 1=STAT_ACCEPTED (meaning the command has been accepted and is being executed – this is the case for “long-running” commands like a periodic measurement), 2..0xF= indicated different error or warning types. Please refer to the datasheet for full details about the error codes.
0x09	PREV_CMD	The last executed command. For information purpose only.
0x20	CID_RID	Configuration ID or Result ID. This is a unique number that identifies the content of the I²C block from 0x21..0xDF. E.g. the common configuration page has a CID=0x16, the measurement result has an RID=0x10, etc.
0x21	TID	A running number that is incremented by the device with every publishing of data on I²C. The host can use it to identify duplicated data records.
0x22	SIZE_LSB	The lower byte of the data size.
0x23	SIZE_MSB	The upper byte of the data size. Note that this byte will always be zero for results, as only raw histograms span more than one I²C packet. Results can be published completely in the I²C registers 0x24..0xDF so there is no need to split the results over multiple packets.
0x24	DATA<0>	First byte of result data/configuration page.
0xDF	DATA<0xbb>	Last byte of possible data/configuration page.

4.1 Configuration Pages

The measurement application is configured via configuration pages. A configuration page contains a set of registers that can be read, modified and written to alter an existing configuration. Below is a simplified drawing of the communication flow:

Figure 10:
Configuration Page Communication Flow



The loading and writing of configuration pages does not trigger an actual action on the device. Only a sanity check is done for a written configuration page. No actual HW reconfiguration is done.

Here is a configuration example for the common configuration page:

1. The measurement period is set to 100 milliseconds,
2. The device is configured to use a different SPAD mask – number 6 and
3. The device drives GPIO0 LOW while the VCSEL is pulsing.

Configuration Steps

1. Load the common configuration page with command LOAD_CONFIG_PAGE_COMMON: S 41 W 08 16 P
2. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as STAT_OK: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)
3. Check that the configuration page is loaded: S 41 W 20 Sr 41 R A A A N P
   This should read back the values: 0x16 <do not care> 0xBC 0x00
4. Change the value of the measurement period to 100ms: S 41 W 24 64 00 P
5. Select pre-defined SPAD mask 6: S 41 W 34 06 P
6. Configure the device for LOW on GPIO0 while the VCSEL is emitting light: S 41 W 31 03 P
7. Write the common page to the device with command WRITE_CONFIG_PAGE: S 41 W 08 15 P
8. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)
9. Enable interrupts for results: Set the register INT_ENAB to 0x02 if you only want to receive result interrupts, set it to 0x62 if you want to receive also error/warning and command done interrupts (see the datasheet for more details): S 41 W E2 02 P or S 41 W E2 62 P
10. Clear any old pending interrupts: S 41 W E1 FF P

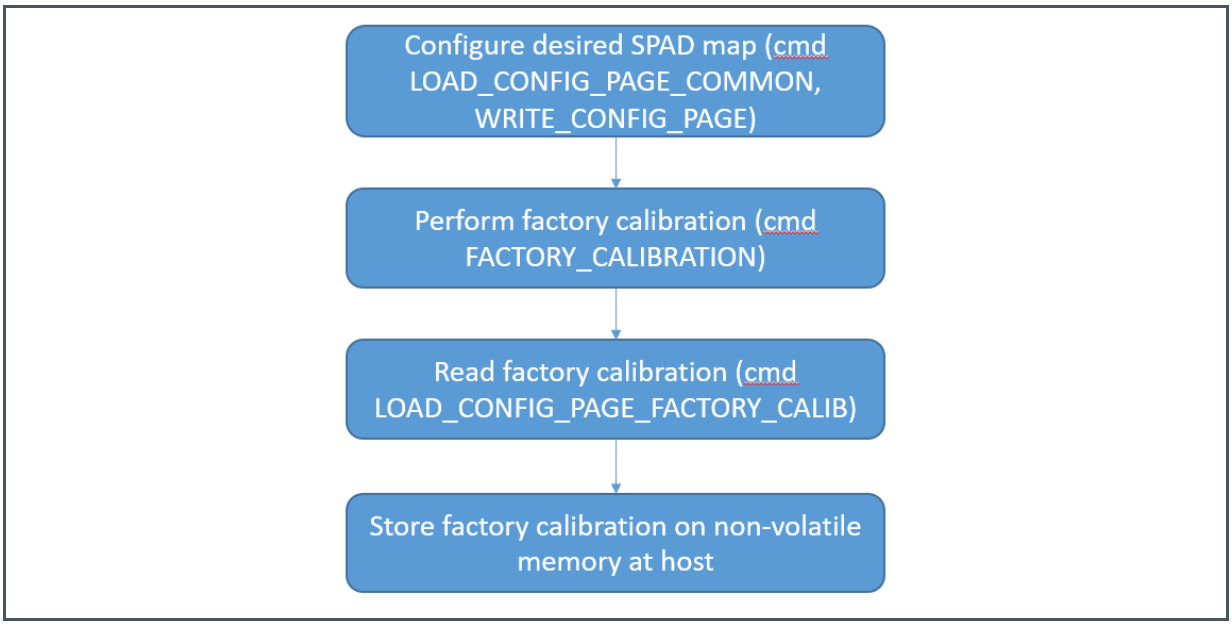
Now the device has been reconfigured internally. The device will perform only simple sanity checks when writing a new configuration.

For the reconfiguration to take effect you need to issue a MEASURE command. The device will perform more checks on your configuration when you issue the MEASURE command.

4.2 Factory Calibration

For the TMF882X to operate at peak performance the device must be factory calibrated together with the final optical stack. Please refer to the **ams** Optical Design Guide for details about the setup for factory calibration.

Figure 11:
Factory Calibration Execution



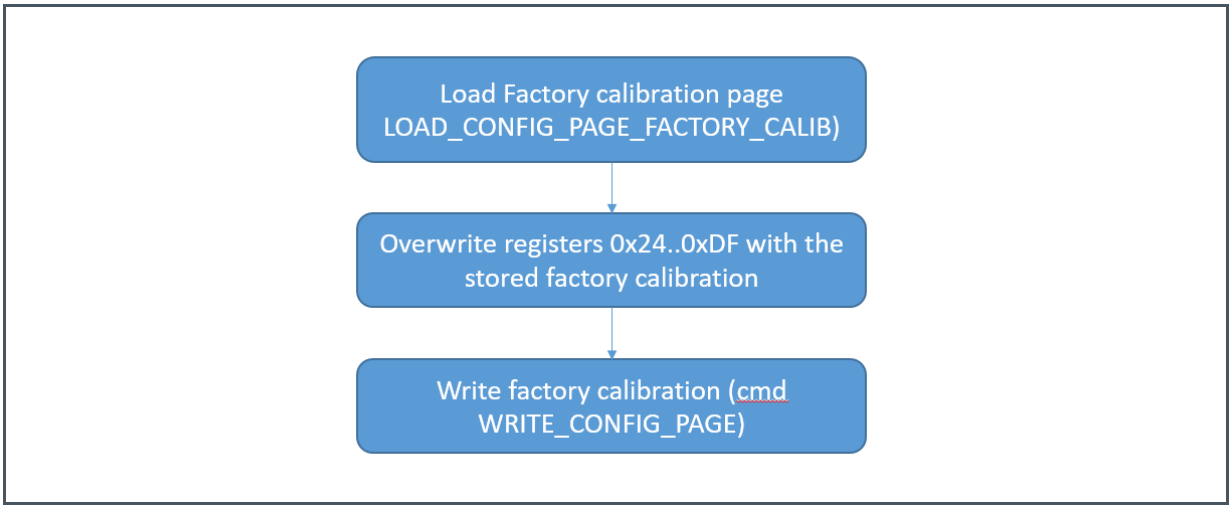
When the device is set up for factory calibration the following sequence of calibration steps should be done:

1. Configure the device through the common configuration page, and make sure the common configuration page is correctly written (see section 4.1).
   Commands: LOAD_CONFIG_PAGE_COMMON and WRITE_CONFIG_PAGE.
2. Initiate factory calibration through command FACTORY_CALIBRATION: S 41 W 08 20 P
3. Wait for the command to terminate by reading the CMD_STAT register until its value is either 0 (STAT_OK=success) or indicates an error (value is in the range of 0x02..0x0F). Note that this command takes some time, so you may read back (depending on your polling interval and I²C speed) 0x01 (STAT_ACCEPTED), which means that the device is currently performing factory calibration. Read back with: S 41 W 08 Sr 41 R N P
4. After successful factory calibration, the host must read out the factory calibration data and store it on the host side for download to the device every time the host has power-cycled the device. To do this, the host must issue the LOAD_CONFIG_PAGE_FACTORY_CALIB command: S 41 W 08 19 P
5. Wait for the command to terminate by reading the CMD_STAT register until its value is either 0 (STAT_OK=success) or indicates an error (value is in the range of 0x02..0x0F).
6. Now the host must read out the complete factory calibration page:
   S 41 W 20 Sr 41 R A A A A …. <repeat> … A N P
   The host must read out the complete factory calibration page from address 0x20 .. 0xDF (inclusive). The factory calibration data itself is located in 0x24..0xDF, the rest is the header for the factory calibration page.

Note that the factory calibration is specific to the selected SPAD map. If a different SPAD map shall be used, the device has to be factory calibrated for this SPAD map, too.

The host is responsible to load the matching factory calibration data to the device for the SPAD map that is shall be used for measurements.

Figure 12:
Factory Calibration Loading



Factory calibration loading means that the host has to do the following steps:

1. Load the factory calibration page with command LOAD_CONFIG_PAGE_FACTORY_CALIB: S 41 W 08 19 P
2. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as STAT_OK: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)
3. Check that the configuration page is loaded: S 41 W 20 Sr 41 R A A A N P
   This should read back the values: 0x19 <do not care> 0xBC 0x00
4. Write the stored calibration data to the I²C registers: 0x24, 0x25, … 0xDF.
5. Write back the calibration data with command WRITE_CONFIG_PAGE: S 41 W 08 15 P
6. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as STAT_OK: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)

Note that if the device does not have a valid factory calibration data or the factory calibration was done for a different SPAD map the device will report a warning in register 0x07 CALIBRATION_STATUS. The value of 0x31 means that no factory calibration has been loaded, the value of 0x32 means that the factory calibration does not match to the selected SPAD map.

Also, note that the loading of factory calibration data does not trigger a comparison of the factory calibration and the selected SPAD map. Only when starting a measurement the firmware will perform this check.

Measurements can be done without factory calibration, in this case the register CALIBRATION_STATUS will report a warning (0x31 or 0x32) and use a default factory calibration. The results without factory calibration data will be less accurate than those of a properly calibrated device.

4.3 Measure Command

After the configuration through the configuration pages and the loading of factory calibration data the device is ready for measurements.

1. Make sure to enable the correct interrupts you want to receive. Enable interrupts for results, set the register INT_ENAB to 0x02 if you only want to receive result interrupts, set it to 0x62 if you want to receive also error/warning and command done interrupts (see the datasheet for more details): S 41 W E2 02 P or S 41 W E2 62 P
2. Clear any old pending interrupts: S 41 W E1 FF P
3. The starting of measurements is done by issuing the command MEASURE: S 41 W 08 10 P
4. The host should check that the command is accepted by reading back the register CMD_STAT: S 41 W 08 Sr 41 R N P. This should read back as 0x01 (if a value >= 0x10 is read back then the host should continue to read this register, if a value < 0x10 and not 0x01 is returned this is an error.)

4.4 Measure Results

When the device has accepted the MEASURE command, the host should wait for the INT pin to be asserted (or poll the register INT_STATUS).

When the INT pin is asserted the host should read out and clear the INT_STATUS register:

Read out and store in variable <int_status> = S 41 W E1 Sr 41 R N P

Clear only the flagged INT_STATUS: S 41 W E1 <int_status> P

Read out the result data with an I²C block read (132 bytes should be read in an I²C block read): S 41 W 20 Sr 41 R A A A … A N P

The result data is in registers: 0x24..0xa3. Please refer to the datasheet for the layout of the result structure.

Information

Note that the results should always be read as an I²C block request, as otherwise it cannot be guaranteed that the read out data is only from one result record. I.e. the device will publish a new result as soon as it is available and the I²C bus is idle. The I²C bus is idle when no host is reading or writing to the I²C bus.

4.5 Measure Stop

The host may stop the device by issuing the command STOP: S 41 W 08 FF P

The host should check that the command is executed successfully by reading back the register CMD_STAT: S 41 W 08 Sr 41 R N P.

This command can never fail unless the device state is corrupted. However, the execution may take up to 2 milliseconds. During this period the firmware tries to shut down the HW gracefully. If this cannot be achieved the firmware will force a shutdown of the HW and return with exit code 0x00 (STAT_OK).

Note that a STOP command (like any other command) may only be issued while the device is not in STANDBY or STANDBY_TIMED state.

To make sure a device is not in standby timed state, please read section 4.6 for more details.

4.6 STANDBY_TIMED

STANDBY_TIMED is a special low power state that the device will only enter if configured to do so, and if the measurement period allows enough time in between the actual measurements to enter this state. The device will enter this special state also only after the host has read out the complete result record for a measurement period.

The host can observe if the device enters STANDBY_TIMED by polling the register ENABLE. IF bit 2 is set, this indicates that the device is currently in state STANDBY_TIMED. Note that the device automatically exits this state when the timer for the measurement period expires.

To stop a device that has been configured for STANDBY_TIMED mode the host has to take the following steps:

Ensure that the device is not in STANDBY_TIMED state, this can be achieved by:

1. Wait for a result interrupt to occur
2. Issue the STOP command: S 41 W 08 FF P
3. Wait for the STOP to be successfully executed by reading CMD_STAT: S 41 W 08 Sr 41 R N P until it has the value 0x00 (STAT_OK):

If the host does not want to wait for an interrupt there is a possible second way:

1. Stop handling the result interrupts (i.e. do not read out results)
2. Perform a wake-up sequence: S 41 W E0 21 P
   This makes sure the device is no longer in STANDBY_TIMED (but the device may still be running).
3. Check that the device is not in STANDBY_TIMED by reading the ENABLE register: S 41 W E0 Sr 41 R N P, which must have the value b_01xx_0001 to continue.
4. Issue the STOP command: S 41 W 08 FF P
5. Wait for the STOP to be successfully executed by reading CMD_STAT: S 41 W 08 Sr 41 R N P until it has the value 0x00 (STAT_OK):

4.7 Histogram Readout

The TMF882X can be configured to provide raw histogram data on I²C. For this mode the device will switch into a blocking mode. I.e. the device will wait for the host to readout a histogram before it proceeds to publish the 2nd time-multiplexed histogram snapshot or the result record.

A raw histogram snapshot of the recorded data consists of:

10x 128x 3 Bytes = 3840 Bytes raw data

This data is organized in packets that are transmitted sequentially via I²C. Each packet has the standard header of 4 bytes, followed by a sub-packet header of 3 bytes, followed by the 128 bytes payload.

Figure 13:
Raw Histogram Packet Format

Absolute I²C Register Address	Name	Description
0x20	RID=0x81	Raw Histogram is published
0x21	TID	Transaction ID, is incremented with every packet
0x22	LSB-SIZE	LSB of total size of this raw histogram, is decremented with each transmitted packet by 0x80 for the payload size.
0x23	MSB-SIZE	MSB of total size. The first packet of a histogram has here 0x0F00
0x24	Sub-packet number	A counter that represents the sub-packet number of the histogram packet. The counter starts with 0 and will count up to 29 (0x1E).
0x25	0x80	Payload of this sub-packet
0x26	0 or 1	The used configuration for the histogram measurement (on TMF8820 devices this will only be 0).
0x27	Data0	First byte of sub-packet
…		
0xA6	Data127	Last byte of sub-packet

Order of transmit of a complete snapshot of raw histograms:

- LSB of channel 0 is transmitted in sub-packet number 0.
- LSB of channel 1 is transmitted in sub-packet number 1.
  …
- LSB of channel 9 is transmitted in sub-packet number 9.
- Mid byte of channel 0 is transmitted in sub-packet number 10.
- Mid byte of channel 1 is transmitted in sub-packet number 11.
  …
- Mid byte of channel 9 is transmitted in sub-packet number 19.
- MSB of channel 0 is transmitted in sub-packet number 20.
- MSB of channel 1 is transmitted in sub-packet number 21.
  …
- MSB of channel 9 is transmitted in sub-packet number 29.

Please note that a complete raw histogram has to be read out before the device will continue.

- The results are published after the corresponding raw histogram(s).
- As the device is in blocking mode, the device will also wait for the host to read out the result data.

- Only after the result has been read, the device will perform the next measurement.

The device has to be configured in the common page to provide histograms. Make sure device is not in STANDBY mode and does not perform measurements. After this do the following steps:

1. Load the common page with command LOAD_CONFIG_PAGE_COMMON:
   S 41 W 08 16 P
2. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as STAT_OK: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)
3. Check that the configuration page is loaded: S 41 W 20 Sr 41 R A A A N P
   This should read back the values: 0x16 <do not care> 0xBC 0x00
4. Write a 0x01 to register 0x39 to dump raw histograms (S 41 W 39 01 P). Write a 0x03 to register 0x39 if you want also to dump electrical calibration histograms (S 41 W 39 03 P).
5. Write back the data with command WRITE_CONFIG_PAGE: S 41 W 08 15 P
6. Check that the command is executed: S 41 W 08 Sr 41 R N P
   This should read back as STAT_OK: 0x00 (if you read back a value >= 0x10 continue to read the register 0x08 until it changes to a value less than 0x10)

Note if you use the INT pin and you have to enable also the raw histogram bit in the INT_ENAB register:

7. Enable the raw histogram (is also the electrical histogram) interrupt if you use the device with the INT pin: S 41 W E2 0A P. Here the assumption is that you also have the result interrupt enabled.

After this normal measurement should be started with: S 41 W 08 10 P

4.7.1 INT Pin Mode

When the INT pin is asserted the host should first read out the register 0xE1 to find out which data is available:

- Result data (bit 1 is set)
- Raw histogram data (bit 3 is set)

I²C string to read out which data is available: S 41 W E1 Sr 41 R N P
The host should store that information.

For measurement results see section 4.4.

For raw histogram readout there are 30 packets to be read for normal mode (TMF8820) and 2x 30 packets for time-multiplexed mode (TMF8821).

The procedure is the same for all 30 packets that form a histogram snapshot:

1. Wait for the raw-histogram interrupt to be flagged in register INT_STATUS (0xE1) – this will also assert the INT pin again.
2. Clear the INT_STATUS register (S 41 W E1 08 P).
3. Read out the complete block of 135 bytes (4 bytes header, 3 bytes sub-header, 128 data payload bytes). Note that the total size of the packet decreases with each readout by 128 bytes and that the sub-packet number increases by 1 for each packet.
4. Repeat steps 1., 2. and 3. for the other 29 packets

After these 30 data packets have been read out by the host, the next interrupt will either be a result interrupt for a normal measurement or another 30 data packets for the 2nd (time-multiplexed) measurement.

The results for time-multiplexed measurements will be published after both raw histograms.

4.7.2 Polling Mode

The same behavior can be achieved by polling the INT_STATUS register only. Here the host does poll the INT_STATUS register until either the result interrupt bit is set or the raw histogram bit is set.

Here the host has to follow the same order of execution:

1. Wait for the raw-histogram interrupt to be flagged in register INT_STATUS (0xE1).
2. Clear the INT_STATUS register.
3. Read out the complete block of 135 bytes (4 bytes header, 3 bytes sub-header, 128 data payload bytes). Note that the total size of the packet decreases with each readout by 128 bytes and that the sub-packet number increases by 1 for each packet.
4. Repeat steps 1., 2. and 3. for the other 29 packets

After these 30 data packets have been read out by the host, the next interrupt will either be a result interrupt for a normal measurement or another 30 data packets for the 2nd (time-multiplexed) measurement.

The results for time-multiplexed measurements will be published after both raw histograms.

5 Revision Information

Changes from previous version to current revision v3-00	Page
Updated all chapters	all
Added histogram readout section, updated for ROM 1v2	all
Typos corrected	all
Change ROMREMAP_RESET to RAMREMAP_RESET	13

- 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.

6 Legal Information

Headquarters
ams AG
Tobelbader Strasse 30
8141 Premstaetten
Austria, Europe
Tel: +43 (0) 3136 500 0

Please visit our website at www.ams.com
Buy our products or get free samples online at www.ams.com/Products
Technical Support is available at www.ams.com/Technical-Support
Provide feedback about this document at www.ams.com/Document-Feedback
For sales offices, distributors and representatives go to www.ams.com/Contact
For further information and requests, e-mail us at ams_sales@ams.com