TMF882X Driver User Guide

Generated by Doxygen 1.8.13

Contents

Chapter 1

TMF882X Driver Overview and Architecture

This document describes the baremetal MCU driver for the Time-of-Flight TMF882X from ams. It is meant to serve as a reference driver using the standard C library without dependencies on an external framework. The core driver modules of are documented with Doxygen.

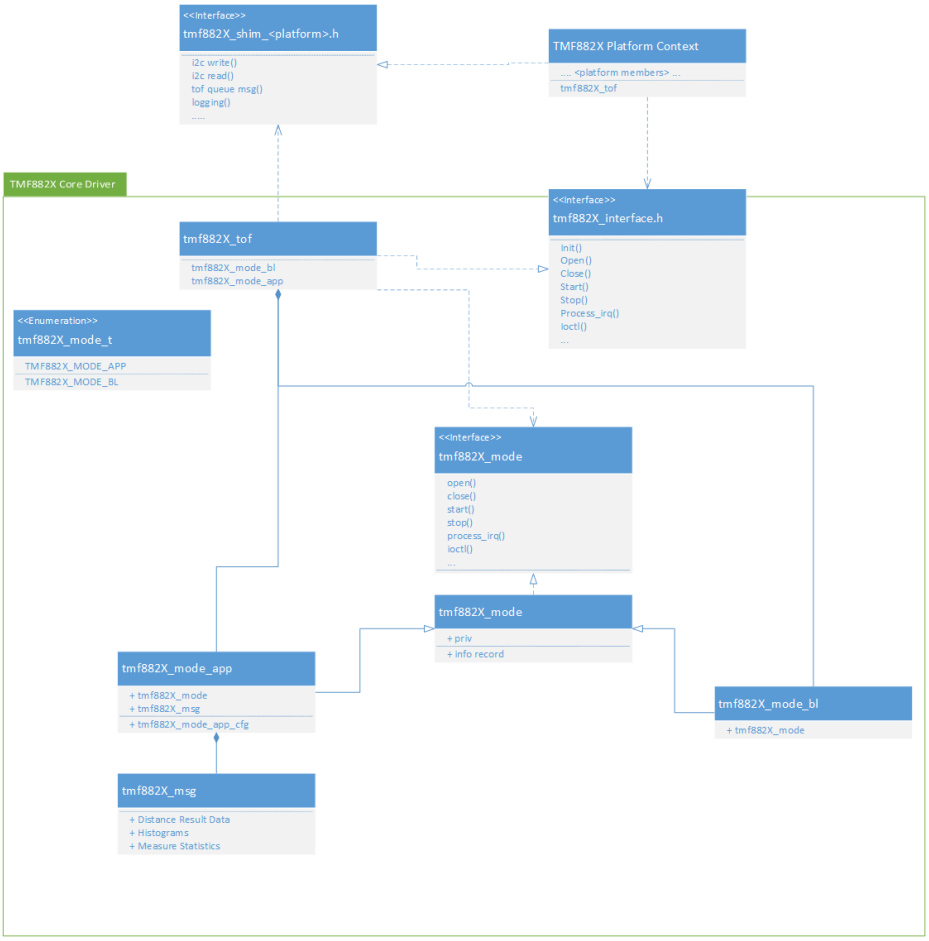
Core Driver Architecture

The TMF882X driver is designed around a "core" modular style driver with a UNIX-like interface using common operations like open, close, ioctl, etc. The core driver client interface is declared in the *tmf882x_interface.h*. The primary structure used in the core driver interface is the core driver context structure tmf882x_tof. The tmf882x_tof structure is used as the first parameter in all of the core driver API functions. The internal definition of the tmf882x_tof structure is exposed in the *tmf882x_interface.h* so that it can be statically declared by the client (though it is not required to be statically allocated).

Below are some of the features and design considerations of the core reference driver:

- The core driver is implemented in C99
- The core driver performs no dynamic memory allocation
- All platform specific functions for hardware access, logging, etc. are handled through a client-defined callback shim layer
- The core driver has no mutual exclusion mechanisms, and the client must handle any synchronization when using the core driver in a multi-threaded application
- The core driver has two log levels, a base level used for informational and error logging and a debug level for more verbose logging

Below is an architecture class diagram of the core driver:



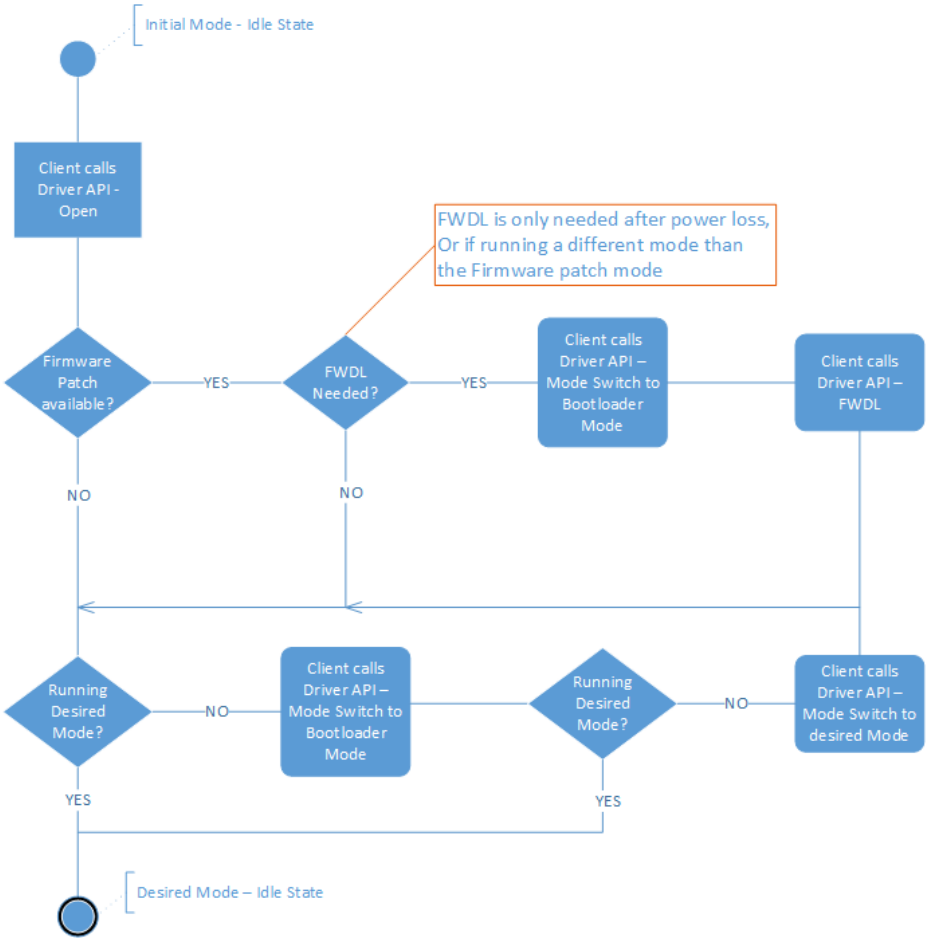
The core driver operates based on the context of the current operational "mode", *tmf882x_mode_t*. Not all modes implement all of the available interface functions defined in *tmf882x_interface.h*, those that are not implemented/supported return an error.

Bootloader Mode

The primary purpose of the Bootloader mode is to help load other operational modes. The mode loading options in the bootloader are currently:

```
1. Switch to mode from Non-Volatile Memory
2. Firmware Download (FWDL) mode
```

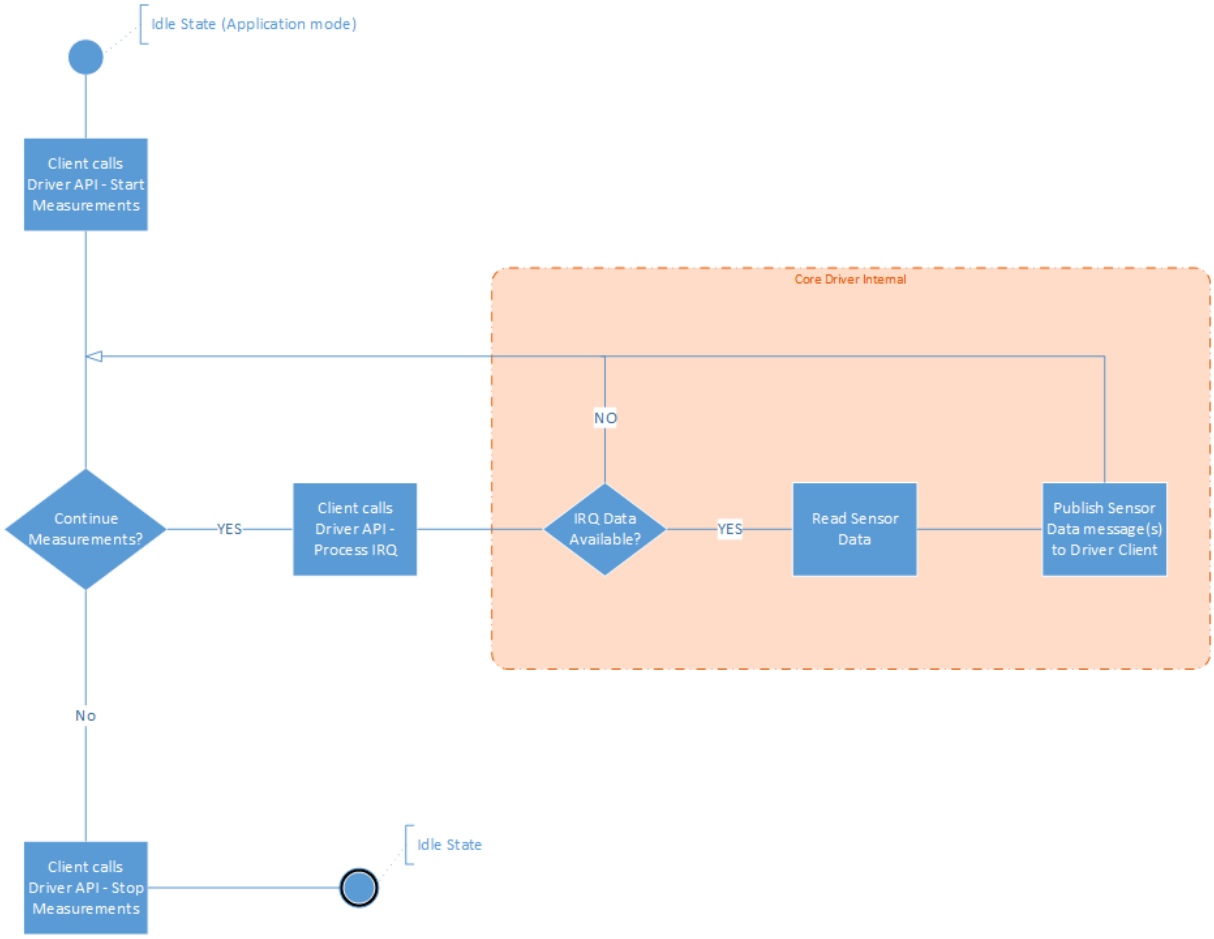
Below is a flowchart diagram of swapping between operational modes with the core driver interface.



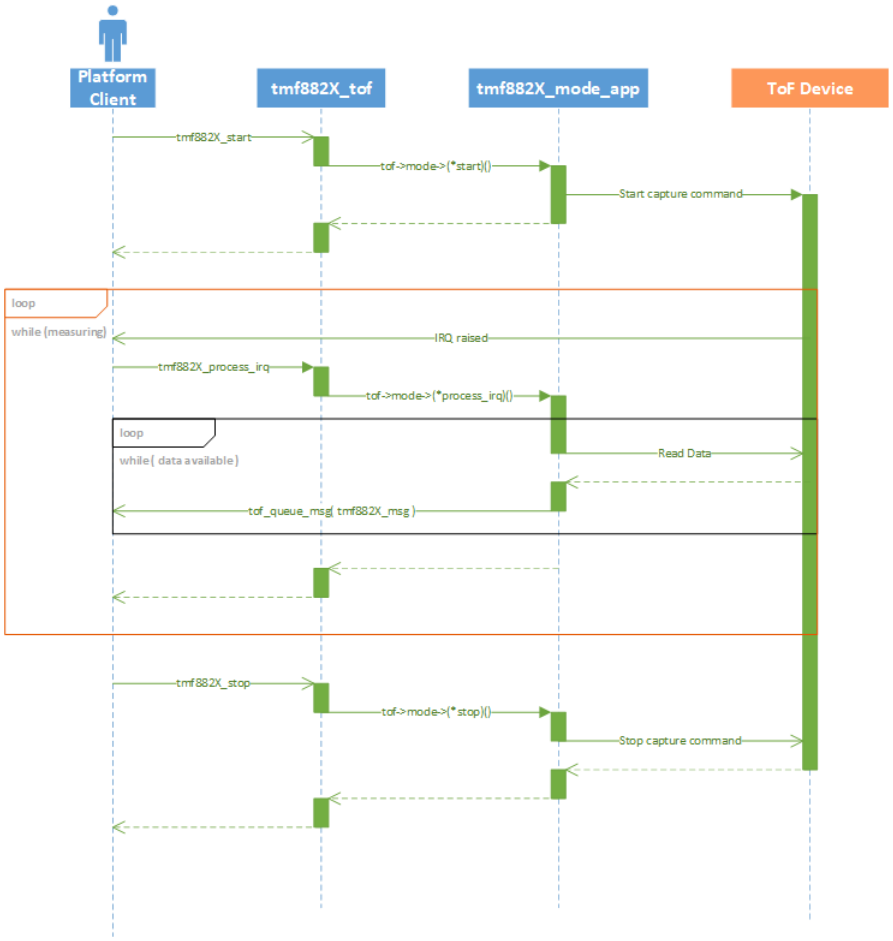
Application Mode

The Application mode is the main operational mode of the TMF882X sensor. This mode is used to perform ToF measurements and return measurement data. The Application mode only supports switching modes back to the Bootloader operational mode.

Below is a measurement flowchart diagram in the Application mode with the core driver interface.



And here is a sequence diagram of the Application mode measurements and data flow within the core driver interface.


Platform Client
tmf882X_tof
tmf882X_mode_app
ToF Device
tmf882X_start
tof->mode->(*start)()
Start capture command
loop
while (measuring)
IRQ raised
tmf882X_process_irq
tof->mode->(*process_irq)()
loop
while(data available)
Read Data
tof_queue_msg(tmf882X_msg)
tmf882X_stop
tof->mode->(*stop)()
Stop capture command

Chapter 2

TMF882X Driver Porting Guide

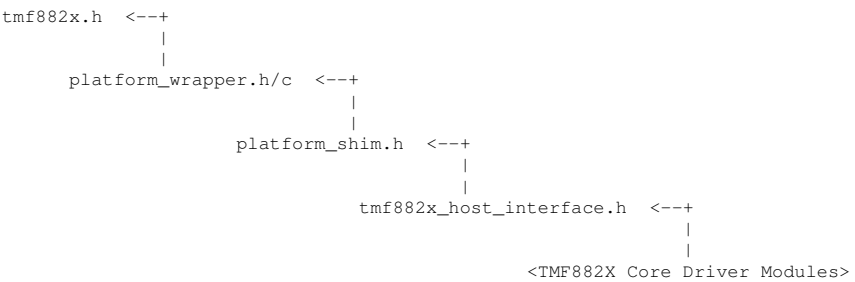
This document gives an overview of the general changes necessary to port the TMF882X MCU driver to other platforms.

Core Driver Interface OS Abstraction Layer (OSAL)

The TMF882X core driver relies on an OSAL layer to abstract I/O, logging, time interfaces, etc. from the core driver functionality.

See the Control / Data Flow graph below:

See the module dependency chain below:



Abstraction Module descriptions:

- tmf882x.h
  - Core driver output data types
  - This is part of the core driver and should not be changed. Client applications can include this file for parsing output data.
- platform_wrapper.c(h)
  - Abstraction layer between core driver and target platform/Client applications
  - purpose of this module is to decouple client applications from the core driver interface. In the reference MCU driver, platform interfaces needed by the core driver in the platform_shim.h layer are implemented here.
  - **This should be re-implemented for porting to new platforms**
- platform_shim.h
  - part of OSAL/HAL abstraction between the tmf882x core driver and target platform
  - all OSAL interface functions in this file must be implemented for the target platform for full functionality. The only exception are logging functions which are optional
  - **This should be re-implemented for porting to new platforms**
- tmf882x_host_interface.h
  - The purpose of this file is to include the target platform interface shim layer. This include is used by the core driver to include the OSAL interface declarations
- TMF882X Core Driver
  - All other files not listed above are considered part of the core driver and should not need to be modified for porting to other target platforms.

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 _intelHexRecord Struct Reference

Public Attributes

- uint32_t **ulba**
- uint32_t **address**
- uint32_t **length**
- uint8_t **data** [INTEL_HEX_MAX_RECORD_DATA_SIZE]

The documentation for this struct was generated from the following file:

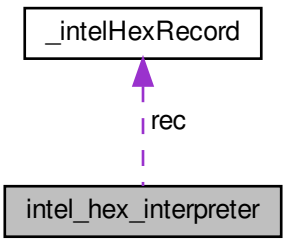
- intel_hex_interpreter.h

5.2 intel_hex_interpreter Struct Reference

This is the Base mode behavioral function pointer structure.

`#include <intel_hex_interpreter.h>`

Collaboration diagram for intel_hex_interpreter:



Public Attributes

- const uint8_t * hex_records
- uint32_t hex_size
- uint32_t count
- uint32_t **last_addr**
- bool **eof_reached**
- intelRecord rec

5.2.1 Detailed Description

This is the Base mode behavioral function pointer structure.

5.2.2 Member Data Documentation

5.2.2.1 count

`intel_hex_interpreter::count`

This member tracks the position of the interpreter in the parser

5.2.2.2 hex_records

`intel_hex_interpreter::hex_records`

This member points to the string buffer containing all of the new-line separated hex records

5.2.2.3 hex_size

`intel_hex_interpreter::hex_size`

This member contains the size of the string in intel_hex_interpreter::hex_records

5.2.2.4 rec

`intel_hex_interpreter::rec`

This member is used as a temporary buffer for parsing individual records

The documentation for this struct was generated from the following file:

- intel_hex_interpreter.h

5.3 intelRecord Struct Reference

This structure represents a single intel hex record.

`#include <intel_hex_interpreter.h>`

5.3.1 Detailed Description

This structure represents a single intel hex record.

The documentation for this struct was generated from the following file:

- intel_hex_interpreter.h

5.4 mode_vtable Struct Reference

This is the Base mode behavioral function pointer structure.

`#include <tmf882x_mode.h>`

Public Attributes

- uint32_t tag
- int32_t(∗ open)(struct tmf882x_mode ∗self)
- int32_t(∗ fwdl)(struct tmf882x_mode ∗self, int32_t fwdl_type, const uint8_t ∗buf, size_t len)
- int32_t(∗ mode_switch)(struct tmf882x_mode ∗self, uint32_t mode)
- int32_t(∗ start)(struct tmf882x_mode ∗self)
- int32_t(∗ stop)(struct tmf882x_mode ∗self)
- int32_t(∗ process_irq)(struct tmf882x_mode ∗self)
- int32_t(∗ **ioctl**)(struct tmf882x_mode ∗self, uint32_t cmd, const void ∗input, void ∗output)
- void(∗ close)(struct tmf882x_mode ∗self)

5.4.1 Detailed Description

This is the Base mode behavioral function pointer structure.

5.4.2 Member Data Documentation

5.4.2.1 close

`mode_vtable::close`

This member is the close method call for the current mode

Warning

These function pointers should never be called directly, always use the tmf882x_tof interface functions.

5.4.2.2 fwdl

`mode_vtable::fwdl`

This member is the fwdl method call for the current mode

5.4.2.3 mode_switch

`mode_vtable::mode_switch`

This member is the mode_switch method call for the current mode

5.4.2.4 open

`mode_vtable::open`

This member is the open method call for the current mode

5.4.2.5 process_irq

`mode_vtable::process_irq`

This member is the process_irq method call for the current mode

5.4.2.6 start

`mode_vtable::start`

This member is the start method call for the current mode

5.4.2.7 stop

`mode_vtable::stop`

This member is the stop method call for the current mode

5.4.2.8 tag

`mode_vtable::tag`

This member is the mode type identifier

The documentation for this struct was generated from the following file:

- tmf882x_mode.h

5.5 tmf882x_anon_resp Struct Reference

Public Attributes

- uint8_t **data** [BL_MSG_CMD_MAX_SIZE]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.6 tmf882x_clk_corr Struct Reference

This is the Context structure for the clock correction machine.

`#include <tmf882x_clock_correction.h>`

Public Attributes

- uint32_t first_ref
- uint32_t last_ref
- uint32_t first_src
- uint32_t last_src
- uint32_t ratio
- uint32_t iratioQ15
- uint32_t count

5.6.1 Detailed Description

This is the Context structure for the clock correction machine.

5.6.2 Member Data Documentation

5.6.2.1 count

`tmf882x_clk_corr::count`

This member contains the current number of pairs added to the mapping

5.6.2.2 first_ref

`tmf882x_clk_corr::first_ref`

This member contains the first reference clock value

5.6.2.3 first_src

`tmf882x_clk_corr::first_src`

This member contains the first source clock value

5.6.2.4 iratioQ15

`tmf882x_clk_corr::iratioQ15`

This member contains the current inverted ratio of source and reference in Q15 fixed point format

5.6.2.5 last_ref

`tmf882x_clk_corr::last_ref`

This member contains the last reference clock value

5.6.2.6 last_src

`tmf882x_clk_corr::last_src`

This member contains the last source clock value

5.6.2.7 ratio

`tmf882x_clk_corr::ratio`

This member contains the expected ratio of reference and source

The documentation for this struct was generated from the following file:

- tmf882x_clock_correction.h

5.7 tmf882x_info_record Struct Reference

This is the Base mode information record data.

`#include <tmf882x_mode.h>`

Public Attributes

- union {
  struct **record** {
  uint8_t **app_id**
  uint8_t **min_ver**
  uint8_t **build_ver**
  uint8_t **patch_ver**
  uint8_t **reserved_4**
  uint8_t **reserved_5**
  uint8_t **reserved_6**
  uint8_t **reserved_7**
  } **record**
  uint8_t **data** [sizeof(struct record)]
  };

5.7.1 Detailed Description

This is the Base mode information record data.

5.7.2 Member Data Documentation

5.7.2.1 data

`tmf882x_info_record::data`

This member is the data buffer of the info record

5.7.2.2 record

`tmf882x_info_record::record`

This member is the record struct of the info record

The documentation for this struct was generated from the following file:

- tmf882x_mode.h

5.8 tmf882x_meas_result Struct Reference

TMF882X measure result This represents an individual target measurement result.

`#include <tmf882x.h>`

Public Attributes

- uint32_t confidence
- uint32_t distance_mm
- uint32_t channel
- uint32_t ch_target_idx
- uint32_t sub_capture

5.8.1 Detailed Description

TMF882X measure result This represents an individual target measurement result.

5.8.2 Member Data Documentation

5.8.2.1 ch_target_idx

`tmf882x_meas_result::ch_target_idx`

indicates target index in a given channel

This is the index of the target detected if the channel detected more than one target.

5.8.2.2 channel

`tmf882x_meas_result::channel`

channel of result

This is the channel that reported the target

5.8.2.3 confidence

`tmf882x_meas_result::confidence`

confidence level 0 .. no confidence, 0xFFFF .. highest confidence

This is the confidence level of the result reported

5.8.2.4 distance_mm

`tmf882x_meas_result::distance_mm`

distance in mm

This is the distance reported in millimeters

5.8.2.5 sub_capture

`tmf882x_meas_result::sub_capture`

indicates which sub-capture of time-multiplexed measurement

This is the time-multiplexed sub_capture index of the channel that detected the target. For non-time-multiplexed measurements this value is zero.

The documentation for this struct was generated from the following file:

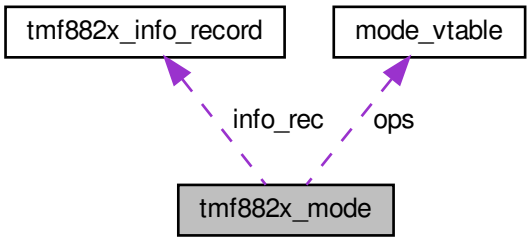
- tmf882x.h

5.9 tmf882x_mode Struct Reference

This is the Base mode context structure.

`#include <tmf882x_mode.h>`

Collaboration diagram for tmf882x_mode:



Public Attributes

- struct mode_vtable const * ops
- struct tmf882x_info_record info_rec
- int32_t debug
- void * priv

5.9.1 Detailed Description

This is the Base mode context structure.

5.9.2 Member Data Documentation

5.9.2.1 debug

`tmf882x_mode::debug`

This member is the debug flag of the current mode

5.9.2.2 info_rec

`tmf882x_mode::info_rec`

This member is the tmf882x_info_record of the current mode

5.9.2.3 ops

`tmf882x_mode::ops`

This member is the mode_vtable of the current mode

5.9.2.4 priv

`tmf882x_mode::priv`

This member is the user-private context of the current mode

The documentation for this struct was generated from the following file:

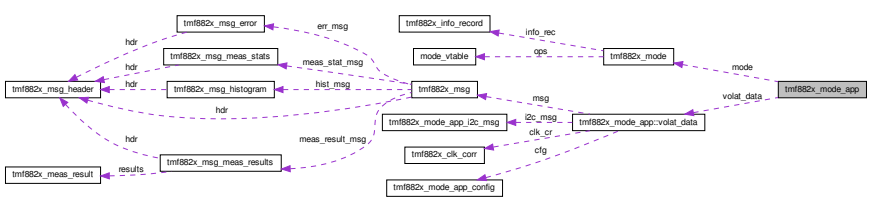
- tmf882x_mode.h

5.10 tmf882x_mode_app Struct Reference

This is the Application mode context structure.

`#include <tmf882x_mode_app.h>`

Collaboration diagram for tmf882x_mode_app:



Classes

- struct volat_data

Public Attributes

- struct tmf882x_mode mode
- struct tmf882x_mode_app::volat_data volat_data

5.10.1 Detailed Description

This is the Application mode context structure.

5.10.2 Member Data Documentation

5.10.2.1 mode

`tmf882x_mode_app::mode`

This member is the tmf882x_mode base class structure

5.10.2.2 volat_data

`tmf882x_mode_app::volat_data`

This member is all of the volatile data within the application mode context structure

The documentation for this struct was generated from the following file:

- tmf882x_mode_app.h

5.11 tmf882x_mode_app_calib Struct Reference

This is the Application mode calibration structure.

`#include <tmf882x_mode_app_ioctl.h>`

Public Attributes

- uint8_t data [TMF882X_MAX_CALIB_SIZE]
- uint32_t calib_len

5.11.1 Detailed Description

This is the Application mode calibration structure.

5.11.2 Member Data Documentation

5.11.2.1 calib_len

`tmf882x_mode_app_calib::calib_len`

Length of calibration data in calibration data buffer

5.11.2.2 data

`tmf882x_mode_app_calib::data`

Calibration data buffer

The documentation for this struct was generated from the following file:

- tmf882x_mode_app_ioctl.h

5.12 tmf882x_mode_app_config Struct Reference

This is the Application mode config structure that holds all configuration parameters for the application.

`#include <tmf882x_mode_app_ioctl.h>`

Public Attributes

- uint16_t report_period_ms
- uint16_t kilo_iterations
- uint16_t low_threshold
- uint16_t high_threshold
- uint32_t zone_mask
- uint8_t persistence
- uint8_t confidence_threshold
- uint8_t gpio_0
- uint8_t gpio_1
- uint8_t power_cfg
- uint8_t spad_map_id
- uint32_t alg_setting
- uint8_t histogram_dump
- uint8_t spread_spectrum
- uint8_t i2c_slave_addr
- uint16_t oscillator_trim

5.12.1 Detailed Description

This is the Application mode config structure that holds all configuration parameters for the application.

5.12.2 Member Data Documentation

5.12.2.1 alg_setting

`tmf882x_mode_app_config::alg_setting`

Algorithm setting configuration

5.12.2.2 confidence_threshold

`tmf882x_mode_app_config::confidence_threshold`

Confidence threshold for generating interrupts

5.12.2.3 gpio_0

`tmf882x_mode_app_config::gpio_0`

GPIO_0 config settings

5.12.2.4 gpio_1

`tmf882x_mode_app_config::gpio_1`

GPIO_1 config settings

5.12.2.5 high_threshold

`tmf882x_mode_app_config::high_threshold`

High distance threshold setting triggering interrupts

5.12.2.6 histogram_dump

`tmf882x_mode_app_config::histogram_dump`

Histogram dump configuration

5.12.2.7 i2c_slave_addr

`tmf882x_mode_app_config::i2c_slave_addr`

I2C slave address configuration

5.12.2.8 kilo_iterations

`tmf882x_mode_app_config::kilo_iterations`

Iterations ∗ 1024 for measurements

5.12.2.9 low_threshold

`tmf882x_mode_app_config::low_threshold`

Low distance threshold setting triggering interrupts

5.12.2.10 oscillator_trim

`tmf882x_mode_app_config::oscillator_trim`

Sensor Oscillator trim value

5.12.2.11 persistence

`tmf882x_mode_app_config::persistence`

Persistence setting for generating interrupts

5.12.2.12 power_cfg

`tmf882x_mode_app_config::power_cfg`

Power configuration settings

5.12.2.13 report_period_ms

`tmf882x_mode_app_config::report_period_ms`

Result reporting period in milliseconds

5.12.2.14 spad_map_id

`tmf882x_mode_app_config::spad_map_id`

Spad map identifier

5.12.2.15 spread_spectrum

`tmf882x_mode_app_config::spread_spectrum`

Spread Spectrum configuration

5.12.2.16 zone_mask

`tmf882x_mode_app_config::zone_mask`

Zone mask for disabling interrupts for certain channels

The documentation for this struct was generated from the following file:

- tmf882x_mode_app_ioctl.h

5.13 tmf882x_mode_app_dev_UID Struct Reference

This is the Application mode structure to hold the device Unique ID.

`#include <tmf882x_mode_app_ioctl.h>`

Public Attributes

- char uid [32]

5.13.1 Detailed Description

This is the Application mode structure to hold the device Unique ID.

5.13.2 Member Data Documentation

5.13.2.1 uid

`tmf882x_mode_app_dev_UID::uid`

Unique Identifier (UID) string

The documentation for this struct was generated from the following file:

- tmf882x_mode_app_ioctl.h

5.14 tmf882x_mode_app_i2c_msg Struct Reference

App mode i2c message.

`#include <tmf882x_mode_app.h>`

Public Attributes

- uint8_t rid
- uint8_t cmd
- uint8_t tid
- uint16_t size
- uint8_t cfg_id
- uint8_t pckt_size
- uint8_t pckt_num
- uint8_t buf [APP_MAX_MSG_SIZE]

5.14.1 Detailed Description

App mode i2c message.

5.14.2 Member Data Documentation

5.14.2.1 buf

`tmf882x_mode_app_i2c_msg::buf`

This member is the message data buffer

5.14.2.2 cfg_id

`tmf882x_mode_app_i2c_msg::cfg_id`

This member is the subcapture configuration ID or breakpoint number

5.14.2.3 cmd

`tmf882x_mode_app_i2c_msg::cmd`

This member is the command message id for sending messages

5.14.2.4 pckt_num

`tmf882x_mode_app_i2c_msg::pckt_num`

This member is the subpacket number in the total message

5.14.2.5 pckt_size

`tmf882x_mode_app_i2c_msg::pckt_size`

This member is the subpacket payload size

5.14.2.6 rid

`tmf882x_mode_app_i2c_msg::rid`

This member is the return message id for receiving messages,

5.14.2.7 size

`tmf882x_mode_app_i2c_msg::size`

This member is the size of the data buffer in the message

5.14.2.8 tid

`tmf882x_mode_app_i2c_msg::tid`

This member is the monotonically-increasing Transaction ID

The documentation for this struct was generated from the following file:

- tmf882x_mode_app.h

5.15 tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config Struct Reference

Public Attributes

- int8_t xoff_q1
- int8_t yoff_q1
- uint8_t xsize
- uint8_t ysize
- uint8_t spad_mask [TMF8X2X_COM_MAX_SPAD_SIZE]
- uint8_t spad_map [TMF8X2X_COM_MAX_SPAD_SIZE]

5.15.1 Member Data Documentation

5.15.1.1 spad_map

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::spad_map`

Spad channel mapping for measurement (channels 1 - 9)

5.15.1.2 spad_mask

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::spad_mask`

Spad enable mask configuration (1 enable, 0 disable)

5.15.1.3 xoff_q1

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::xoff_q1`

X-direction offset in Q1 format

5.15.1.4 xsize

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::xsize`

Size of spad map in X-direction

5.15.1.5 yoff_q1

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::yoff_q1`

Y-direction offset in Q1 format

5.15.1.6 ysize

`tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config::ysize`

Size of spad map in Y-direction

The documentation for this struct was generated from the following file:

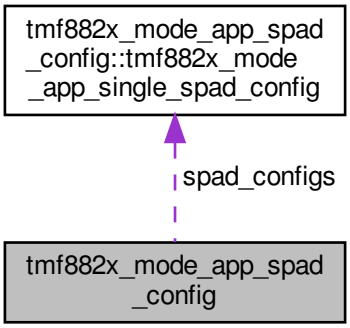
- tmf882x_mode_app_ioctl.h

5.16 tmf882x_mode_app_spad_config Struct Reference

This is the Application mode spad config structure that holds the complete spad configuration for the application.

`#include <tmf882x_mode_app_ioctl.h>`

Collaboration diagram for tmf882x_mode_app_spad_config:



Classes

- struct tmf882x_mode_app_single_spad_config

Public Attributes

- struct tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config spad_configs [TMF8X2↩ X_MAX_CONFIGURATIONS]
- uint32_t num_spad_configs

5.16.1 Detailed Description

This is the Application mode spad config structure that holds the complete spad configuration for the application.

5.16.2 Member Data Documentation

5.16.2.1 num_spad_configs

`tmf882x_mode_app_spad_config::num_spad_configs`

The number of spad configurations in tmf882x_mode_app_spad_configs::spad_configs

Note

The spad enable mask and map size should be 'ysize' * 'xsize' in length

5.16.2.2 spad_configs

`tmf882x_mode_app_spad_config::spad_configs`

The list of spad configurations

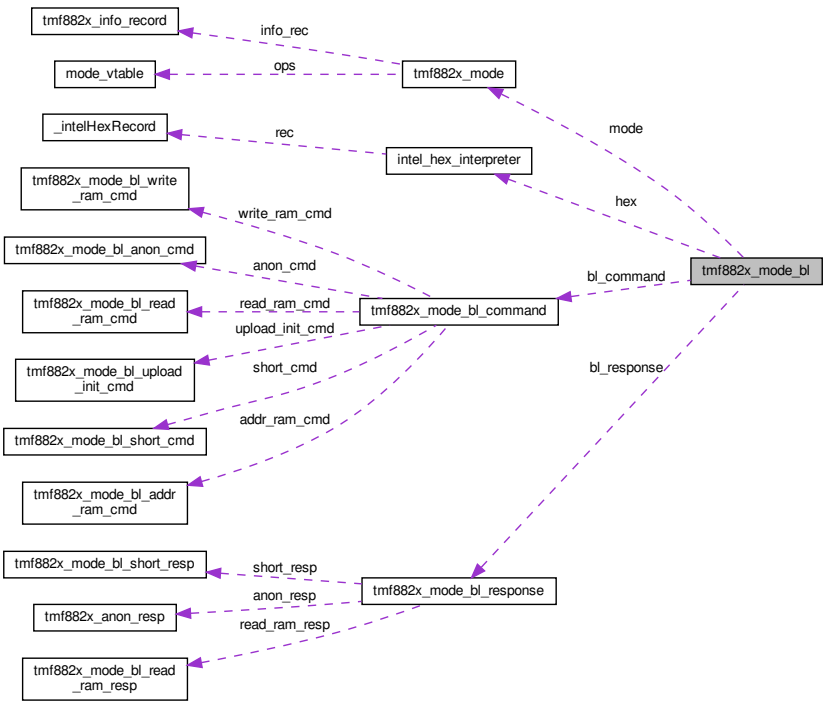
The documentation for this struct was generated from the following file:

- tmf882x_mode_app_ioctl.h

5.17 tmf882x_mode_bl Struct Reference

`#include <tmf882x_mode_bl.h>`

Collaboration diagram for tmf882x_mode_bl:



Public Attributes

- struct tmf882x_mode mode
- struct intel_hex_interpreter hex
- union tmf882x_mode_bl_command bl_command
- union tmf882x_mode_bl_response bl_response

5.17.1 Detailed Description

This is the Bootloader mode context structure

5.17.2 Member Data Documentation

5.17.2.1 bl_command

```
union tmf882x_mode_bl_command tmf882x_mode_bl::bl_command
```

This member is the bootloader command

5.17.2.2 bl_response

```
union tmf882x_mode_bl_response tmf882x_mode_bl::bl_response
```

This member is the bootloader command response

5.17.2.3 hex

```
struct intel_hex_interpreter tmf882x_mode_bl::hex
```

This member is the Intel Hex Interpreter context

5.17.2.4 mode

```
struct tmf882x_mode tmf882x_mode_bl::mode
```

This member is the Base mode context

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.18 tmf882x_mode_bl_addr_ram_cmd Struct Reference

Public Attributes

- uint8_t **command**
- uint8_t **size**
- uint8_t **addr_lsb**
- uint8_t **addr_msb**
- uint8_t **chksum**
- uint8_t **reserved** [BL_MAX_DATA_SZ - 2]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.19 tmf882x_mode_bl_anon_cmd Struct Reference

Public Attributes

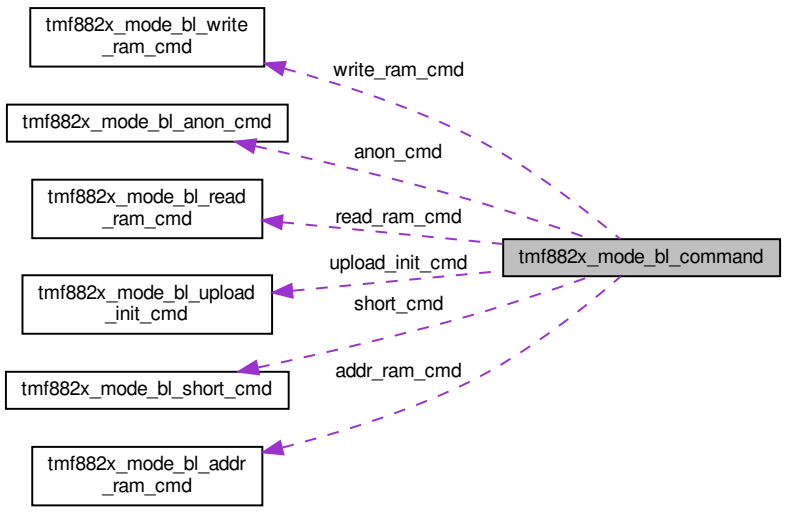
- uint8_t **data** [BL_MSG_CMD_MAX_SIZE]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.20 tmf882x_mode_bl_command Union Reference

Collaboration diagram for tmf882x_mode_bl_command:



Public Attributes

- struct tmf882x_mode_bl_anon_cmd **anon_cmd**
- struct tmf882x_mode_bl_addr_ram_cmd **addr_ram_cmd**
- struct tmf882x_mode_bl_write_ram_cmd **write_ram_cmd**
- struct tmf882x_mode_bl_read_ram_cmd **read_ram_cmd**
- struct tmf882x_mode_bl_upload_init_cmd **upload_init_cmd**
- struct tmf882x_mode_bl_short_cmd **short_cmd**

The documentation for this union was generated from the following file:

- tmf882x_mode_bl.h

5.21 tmf882x_mode_bl_read_ram_cmd Struct Reference

Public Attributes

- uint8_t **command**
- uint8_t **size**
- uint8_t **num_bytes**
- uint8_t **chksum**
- uint8_t **reserved** [BL_MAX_DATA_SZ - 1]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.22 tmf882x_mode_bl_read_ram_resp Struct Reference

Public Attributes

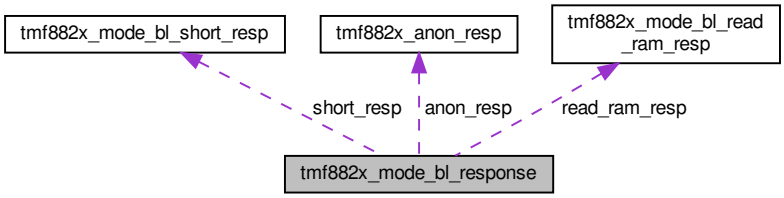
- uint8_t **status**
- uint8_t **size**
- uint8_t **data** [BL_MSG_CMD_MAX_SIZE+1]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.23 tmf882x_mode_bl_response Union Reference

Collaboration diagram for tmf882x_mode_bl_response:



Public Attributes

- struct tmf882x_anon_resp **anon_resp**
- struct tmf882x_mode_bl_read_ram_resp **read_ram_resp**
- struct tmf882x_mode_bl_short_resp **short_resp**

The documentation for this union was generated from the following file:

- tmf882x_mode_bl.h

5.24 tmf882x_mode_bl_short_cmd Struct Reference

Public Attributes

- uint8_t **command**
- uint8_t **size**
- uint8_t **chksum**
- uint8_t **reserved** [BL_MAX_DATA_SZ]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.25 tmf882x_mode_bl_short_resp Struct Reference

Public Attributes

- uint8_t **status**
- uint8_t **size**
- uint8_t **reserved** [BL_MSG_CMD_MAX_SIZE - 3]
- uint8_t **chksum**

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.26 tmf882x_mode_bl_upload_init_cmd Struct Reference

Public Attributes

- uint8_t **command**
- uint8_t **size**
- uint8_t **seed**
- uint8_t **chksum**
- uint8_t **reserved** [BL_MAX_DATA_SZ - 1]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.27 tmf882x_mode_bl_write_ram_cmd Struct Reference

Public Attributes

- uint8_t **command**
- uint8_t **size**
- uint8_t **data** [BL_MAX_DATA_SZ+1]

The documentation for this struct was generated from the following file:

- tmf882x_mode_bl.h

5.28 tmf882x_msg Struct Reference

TMF882X message type. This is the global generic message type returned by the core driver.

`#include <tmf882x.h>`

Collaboration diagram for tmf882x_msg:

Public Attributes

- union {
  struct tmf882x_msg_header **hdr**
  struct tmf882x_msg_error **err_msg**
  struct tmf882x_msg_histogram **hist_msg**
  struct tmf882x_msg_meas_results **meas_result_msg**
  struct tmf882x_msg_meas_stats **meas_stat_msg**
  uint8_t **msg_buf** [TMF882X_MAX_MSG_SIZE]
  };

5.28.1 Detailed Description

TMF882X message type. This is the global generic message type returned by the core driver.

5.28.2 Member Data Documentation

5.28.2.1 err_msg

`tmf882x_msg::err_msg`

This is the error message struct tmf882x_msg_error

5.28.2.2 hdr

`tmf882x_msg::hdr`

This is the message header struct tmf882x_msg_header

5.28.2.3 hist_msg

`tmf882x_msg::hist_msg`

This is the histogram message struct tmf882x_msg_histogram

5.28.2.4 meas_result_msg

`tmf882x_msg::meas_result_msg`

This is the results message struct tmf882x_msg_meas_results

5.28.2.5 meas_stat_msg

`tmf882x_msg::meas_stat_msg`

This is the statistics message struct tmf882x_msg_meas_stats

5.28.2.6 msg_buf

`tmf882x_msg::msg_buf`

This is the low level buffer used to hold the message

The documentation for this struct was generated from the following file:

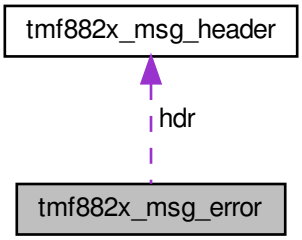
- tmf882x.h

5.29 tmf882x_msg_error Struct Reference

TMF882X error message type. This error message is returned by the core driver for error such as communication errors.

`#include <tmf882x.h>`

Collaboration diagram for tmf882x_msg_error:



Public Attributes

- struct tmf882x_msg_header hdr
- uint32_t err_code

5.29.1 Detailed Description

TMF882X error message type. This error message is returned by the core driver for error such as communication errors.

5.29.2 Member Data Documentation

5.29.2.1 err_code

`tmf882x_msg_error::err_code`

This is the error code identifier

5.29.2.2 hdr

`tmf882x_msg_error::hdr`

This is the error message header struct tmf882x_msg_header

The documentation for this struct was generated from the following file:

- tmf882x.h

5.30 tmf882x_msg_header Struct Reference

TMF882X message header type.

`#include <tmf882x.h>`

Public Attributes

- uint32_t msg_id
- uint32_t msg_len

5.30.1 Detailed Description

TMF882X message header type.

5.30.2 Member Data Documentation

5.30.2.1 msg_id

`tmf882x_msg_header::msg_id`

This member holds the message identifier code

5.30.2.2 msg_len

`tmf882x_msg_header::msg_len`

This member holds the message length (including the header)

The documentation for this struct was generated from the following file:

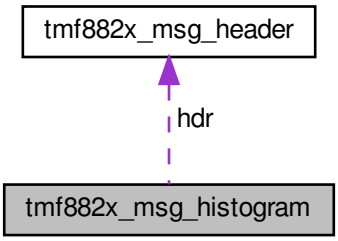
- tmf882x.h

5.31 tmf882x_msg_histogram Struct Reference

TMF882X histogram message type. This message is returned by the core driver for each set of historams that are received by the core driver from the device.

`#include <tmf882x.h>`

Collaboration diagram for tmf882x_msg_histogram:



Public Attributes

- struct tmf882x_msg_header hdr
- uint32_t capture_num
- uint32_t sub_capture
- uint32_t histogram_type
- uint32_t num_tdc
- uint32_t num_bins
- uint32_t bins [TMF882X_HIST_NUM_TDC][TMF882X_HIST_NUM_BINS]

5.31.1 Detailed Description

TMF882X histogram message type. This message is returned by the core driver for each set of historams that are received by the core driver from the device.

5.31.2 Member Data Documentation

5.31.2.1 bins

`tmf882x_msg_histogram::bins`

These are the histogram bin values for each TDC. There are two channels per TDC, the first channel histogram occupies bins [0 : TMF882X_HIST_NUM_BINS/2 - 1], the 2nd channel occupies bins [TMF882X_HIST_NUM_BINS/2 : TMF882X_HIST_NUM_BINS - 1]

5.31.2.2 capture_num

`tmf882x_msg_histogram::capture_num`

This is the capture number that this set of histograms is associated with. The capture_num will match the struct tmf882x_msg_meas_results::result_num

5.31.2.3 hdr

`tmf882x_msg_histogram::hdr`

This is the message header struct tmf882x_msg_header

5.31.2.4 histogram_type

`tmf882x_msg_histogram::histogram_type`

This is the histogram type identifier code enum tmf882x_histogram_type

5.31.2.5 num_bins

`tmf882x_msg_histogram::num_bins`

This is the number of bins in the histograms being published

5.31.2.6 num_tdc

`tmf882x_msg_histogram::num_tdc`

This is the number of TDC histograms being published

5.31.2.7 sub_capture

`tmf882x_msg_histogram::sub_capture`

This is the time-multiplexed sub-capture index of this set of histograms. For non-time-multiplexed measurements this value is always zero.

The documentation for this struct was generated from the following file:

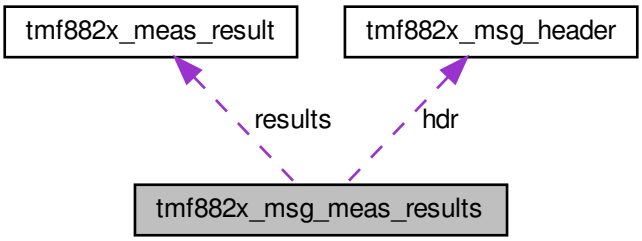
- tmf882x.h

5.32 tmf882x_msg_meas_results Struct Reference

TMF882X measure results message type. This message is returned by the core driver for each set of measurement results that are received by the core driver from the device.

`#include <tmf882x.h>`

Collaboration diagram for tmf882x_msg_meas_results:



Public Attributes

- struct tmf882x_msg_header hdr
- uint32_t result_num
- uint32_t temperature
- uint32_t ambient_light
- uint32_t photon_count
- uint32_t ref_photon_count
- uint32_t sys_ticks
- uint32_t valid_results
- uint32_t num_results
- struct tmf882x_meas_result results [TMF882X_MAX_MEAS_RESULTS]

5.32.1 Detailed Description

TMF882X measure results message type. This message is returned by the core driver for each set of measurement results that are received by the core driver from the device.

5.32.2 Member Data Documentation

5.32.2.1 ambient_light

`tmf882x_msg_meas_results::ambient_light`

This is the ambient light level reported by the device

5.32.2.2 hdr

`tmf882x_msg_meas_results::hdr`

This is the message header struct tmf882x_msg_header

5.32.2.3 num_results

`tmf882x_msg_meas_results::num_results`

This is the number of non-zero targets counted by the core driver

5.32.2.4 photon_count

`tmf882x_msg_meas_results::photon_count`

This is the photon count reported by the device

5.32.2.5 ref_photon_count

`tmf882x_msg_meas_results::ref_photon_count`

This is the reference channel photon count reported by the device

5.32.2.6 result_num

`tmf882x_msg_meas_results::result_num`

This is the result number reported by the device

5.32.2.7 results

`tmf882x_msg_meas_results::results`

This is the list of measurement targets struct tmf882x_meas_result

5.32.2.8 sys_ticks

`tmf882x_msg_meas_results::sys_ticks`

This is the system tick counter (5MHz counter) reported by the device. This is used by the core driver to perform clock compensation correction on the measurement results.

5.32.2.9 temperature

`tmf882x_msg_meas_results::temperature`

This is the temperature reported by the device (in Celsius)

5.32.2.10 valid_results

`tmf882x_msg_meas_results::valid_results`

This is the number of targets reported by the device

The documentation for this struct was generated from the following file:

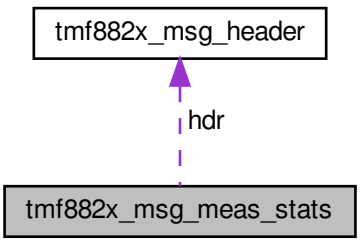
- tmf882x.h

5.33 tmf882x_msg_meas_stats Struct Reference

TMF882X measure statistics message type. This message is returned by the core driver for each set of measurement statistics that are received by the core driver from the device.

`#include <tmf882x.h>`

Collaboration diagram for tmf882x_msg_meas_stats:



Public Attributes

- struct tmf882x_msg_header hdr
- uint32_t capture_num
- uint32_t sub_capture
- uint32_t tdcif_status
- uint32_t iterations_configured
- uint32_t remaining_iterations
- uint32_t accumulated_hits
- uint32_t raw_hits [TMF882X_HIST_NUM_TDC]
- uint32_t saturation_cnt [TMF882X_HIST_NUM_TDC]

5.33.1 Detailed Description

TMF882X measure statistics message type. This message is returned by the core driver for each set of measurement statistics that are received by the core driver from the device.

5.33.2 Member Data Documentation

5.33.2.1 accumulated_hits

`tmf882x_msg_meas_stats::accumulated_hits`

This is the accumulated hits reported by the device

5.33.2.2 capture_num

`tmf882x_msg_meas_stats::capture_num`

This is the capture number that this set of statistics is associated with. The capture_num will match the struct tmf882x_msg_meas_results::result_num

5.33.2.3 hdr

`tmf882x_msg_meas_stats::hdr`

This is the message header struct tmf882x_msg_header

5.33.2.4 iterations_configured

`tmf882x_msg_meas_stats::iterations_configured`

This is the iterations configured reported by the device

5.33.2.5 raw_hits

`tmf882x_msg_meas_stats::raw_hits`

This is the raw hits reported by the device for each TDC

5.33.2.6 remaining_iterations

`tmf882x_msg_meas_stats::remaining_iterations`

This is the remaining iterations reported by the device

5.33.2.7 saturation_cnt

`tmf882x_msg_meas_stats::saturation_cnt`

This is the saturation count reported by the device for each TDC

5.33.2.8 sub_capture

`tmf882x_msg_meas_stats::sub_capture`

This is the time-multiplexed sub_capture index of the channel that detected the target. For non-time-multiplexed measurements this value is zero.

5.33.2.9 tdcif_status

`tmf882x_msg_meas_stats::tdcif_status`

This is the tdcif status reported by the device

The documentation for this struct was generated from the following file:

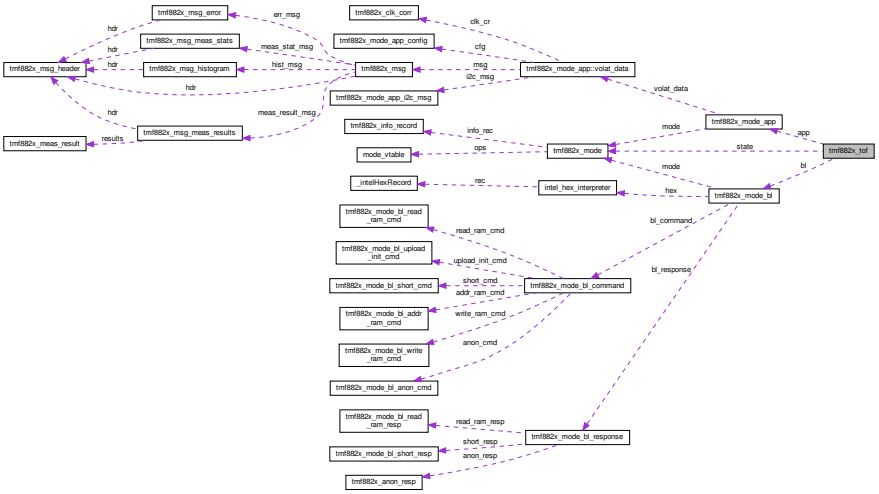
- tmf882x.h

5.34 tmf882x_tof Struct Reference

TMF882X DCB context handle.

`#include <tmf882x_interface.h>`

Collaboration diagram for tmf882x_tof:



Public Attributes

- union {
  struct tmf882x_mode_bl **bl**
  struct tmf882x_mode_app **app**
  struct tmf882x_mode **state**
  };

5.34.1 Detailed Description

TMF882X DCB context handle.

5.34.2 Member Data Documentation

5.34.2.1 app

`tmf882x_tof::app`

This member holds the application state context

5.34.2.2 bl

`tmf882x_tof::bl`

This member holds the bootloader state context

5.34.2.3 state

`tmf882x_tof::state`

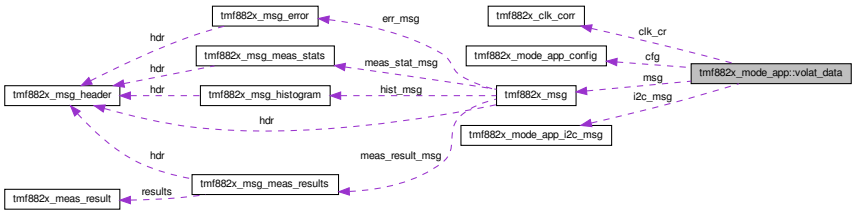
This member holds the base state context

The documentation for this struct was generated from the following file:

- tmf882x_interface.h

5.35 tmf882x_mode_app::volat_data Struct Reference

Collaboration diagram for tmf882x_mode_app::volat_data:



Public Attributes

- bool is_open
- bool is_measuring
- bool clk_corr_enabled
- uint32_t irq
- struct tmf882x_clk_corr **clk_cr**
- struct tmf882x_msg msg
- struct tmf882x_mode_app_i2c_msg i2c_msg
- uint32_t capture_num
- struct tmf882x_mode_app_config cfg
- uint8_t uid [sizeof(uint32_t)]
- struct timespec timestamp

5.35.1 Member Data Documentation

5.35.1.1 capture_num

`tmf882x_mode_app::volat_data::capture_num`

This member is the monotonically-increasing capture number for each result

5.35.1.2 cfg

`tmf882x_mode_app::volat_data::cfg`

This member is the tmf882x_mode_app_config configuration used for writing/reading configuration from the application mode. Two configuration structure tables are supported by the device

5.35.1.3 clk_corr_enabled

`tmf882x_mode_app::volat_data::clk_corr_enabled`

This member is whether the application mode is compensating results for clock skew

5.35.1.4 i2c_msg

`tmf882x_mode_app::volat_data::i2c_msg`

This member is the tmf882x_mode_app_i2c_msg for sending/receiving i2c messages from the application mode

5.35.1.5 irq

`tmf882x_mode_app::volat_data::irq`

This member is the cached IRQ status while servicing device interrupts

5.35.1.6 is_measuring

`tmf882x_mode_app::volat_data::is_measuring`

This member is whether the application mode is currently measuring

5.35.1.7 is_open

`tmf882x_mode_app::volat_data::is_open`

This member is whether the application mode is open

5.35.1.8 msg

`tmf882x_mode_app::volat_data::msg`

This member is the tmf882x_msg for return data output from the device

5.35.1.9 timestamp

`tmf882x_mode_app::volat_data::timestamp`

This member is the cached previous timestamp used in clock correction

5.35.1.10 uid

`tmf882x_mode_app::volat_data::uid`

Buffer for reading out the Device UID

The documentation for this struct was generated from the following file:

- tmf882x_mode_app.h

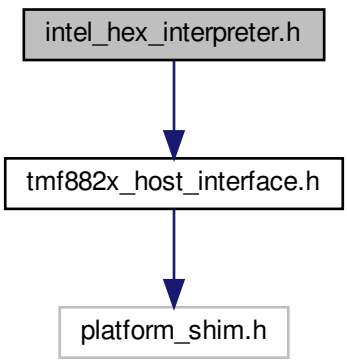
Chapter 6

File Documentation

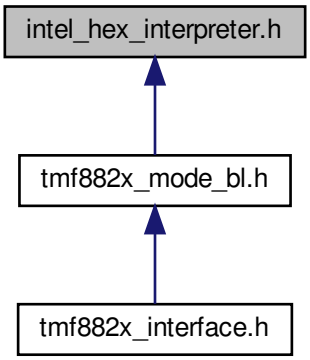
6.1 intel_hex_interpreter.h File Reference

```
#include "tmf882x_host_interface.h"
```

Include dependency graph for intel_hex_interpreter.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct _intelHexRecord
- struct intel_hex_interpreter

  This is the Base mode behavioral function pointer structure.

Macros

- #define **INTEL_HEX_TYPE_DATA** 0
- #define **INTEL_HEX_TYPE_EOF** 1
- #define **INTEL_HEX_TYPE_EXT_LIN_ADDR** 4
- #define **INTEL_HEX_TYPE_START_LIN_ADDR** 5
- #define **INTEL_HEX_EOF** 1 /∗ end of file -> reset ∗/
- #define **INTEL_HEX_CONTINUE** 0 /∗ continue reading in ∗/
- #define **INTEL_HEX_ERR_NOT_A_NUMBER** -1
- #define **INTEL_HEX_ERR_TOO_SHORT** -2
- #define **INTEL_HEX_ERR_CRC_ERR** -3
- #define **INTEL_HEX_ERR_UNKNOWN_TYPE** -4
- #define **INTEL_HEX_WRITE_FAILED** -5
- #define **INTEL_HEX_ULBA**(adr) ((adr) & 0xFFFF0000UL)
- #define **INTEL_HEX_MIN_RECORD_SIZE** 11
- #define **INTEL_HEX_MIN_LAST_ADDRESS** ((INTEL_HEX_MIN_RECORD_SIZE) - 1)
- #define **INTEL_HEX_MAX_RECORD_DATA_SIZE** (128)

Typedefs

- typedef struct _intelHexRecord **intelRecord**

Functions

- void ihexi_init (struct intel_hex_interpreter *hex, const uint8_t *hex_records, uint32_t size)

  Initialize an intel hex interpreter.
- int32_t ihexi_get_next_bin (struct intel_hex_interpreter *hex, uint8_t *buf, uint32_t length, uint32_t *addr)

  Parses an Intel Hex Record file records at a time and outputs binary data blobs.
- bool ihexi_is_eof (struct intel_hex_interpreter *hex)

  Return whether EOF record was reached by the parser.

6.1.1 Function Documentation

6.1.1.1 ihexi_get_next_bin()

```
int32_t ihexi_get_next_bin (
            struct intel_hex_interpreter * hex,
            uint8_t * buf,
            uint32_t length,
            uint32_t * addr )
```

Parses an Intel Hex Record file records at a time and outputs binary data blobs.

Parameters

in	*hex*	pointer to string of Intel Hex Records
out	*buf*	buffer to place data blob
in	*length*	size of buf
out	*addr*	address of data blob in buf

Returns

number of Bytes copied to buf, negative if error occurs, 0 indicates EOF

6.1.1.2 ihexi_init()

```
void ihexi_init (
            struct intel_hex_interpreter * hex,
            const uint8_t * hex_records,
            uint32_t size )
```

Initialize an intel hex interpreter.

Parameters

in	*hex*	pointer to intel hex interpreter context structure
in	*hex_records*	pointer to string of hex to decode
in	*size*	size of hex_records buffer

6.1.1.3 ihexi_is_eof()

```
bool ihexi_is_eof (
            struct intel_hex_interpreter * hex )
```

Return whether EOF record was reached by the parser.

Parameters

in	*hex*	pointer to intel hex interpreter context structure

Note

This function should always be called by the client to verify a complete set of hex records have been parsed

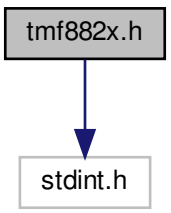
Returns

bool Hex record(s) EOF status

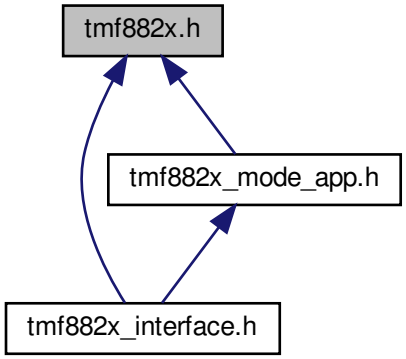
6.2 tmf882x.h File Reference

```
#include <stdint.h>
```

Include dependency graph for tmf882x.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tmf882x_msg_header

  TMF882X message header type.
- struct tmf882x_msg_error

  TMF882X error message type. This error message is returned by the core driver for error such as communication errors.
- struct tmf882x_msg_histogram

  TMF882X histogram message type. This message is returned by the core driver for each set of historams that are received by the core driver from the device.
- struct tmf882x_meas_result

  TMF882X measure result This represents an individual target measurement result.
- struct tmf882x_msg_meas_results

  TMF882X measure results message type. This message is returned by the core driver for each set of measurement results that are received by the core driver from the device.
- struct tmf882x_msg_meas_stats

  TMF882X measure statistics message type. This message is returned by the core driver for each set of measurement statistics that are received by the core driver from the device.
- struct tmf882x_msg

  TMF882X message type. This is the global generic message type returned by the core driver.

Macros

- #define TMF882X_HIST_NUM_BINS 256

  Number of histogram bins.
- #define TMF882X_HIST_NUM_TDC 5

  Number of TDCs.
- #define TMF882X_BYTES_PER_BIN 4

  Number of Bytes per histogram bin.
- #define TMF882X_MAX_MEAS_RESULTS 36

  Maximum number of measurement result targets that can be reported at once.

- #define TMF882X_NUM_CH_PER_TDC 2

  Number of channels per TDC.
- #define TMF882X_NUM_CH ((TMF882X_HIST_NUM_TDC)*(TMF882X_NUM_CH_PER_TDC))

  Total Number of channels.
- #define TMF882X_MAX_MSG_SIZE

  Maximum message size.
- #define **TMF882X_MSG_HEADER_SIZE** sizeof(struct tmf882x_msg_header)
- #define **TOF_ZERO_MSG**(msg)
- #define **TOF_INIT_MSG**(msg)
- #define **TOF_SET_MSG_HDR**(msg, id, type)
- #define **TOF_SET_ERR_MSG**(msg, errid)
- #define **TOF_SET_HISTOGRAM_MSG**(msg, hist_type)

Enumerations

- enum tmf882x_msg_id { **ID_MEAS_RESULTS** = 0x01, **ID_MEAS_STATS** = 0x02, **ID_HISTOGRAM** = 0x03, **ID_ERROR** = 0x0F }

  Output message identifier codes.
- enum tmf882x_histogram_type { **HIST_TYPE_RAW** = 0, **HIST_TYPE_ELEC_CAL** = 1, **TMF882X_NUM_HIST_TYPES** }

  Histogram message type identifier codes.
- enum tmf882x_msg_error_codes { **ERR_COMM** = 0xFD, **ERR_RETRIES** = 0xFE, **ERR_BUF_OVERFLOW** = 0xFF }

  Error message type identifier codes.

6.2.1 Macro Definition Documentation

6.2.1.1 TMF882X_MAX_MSG_SIZE

```
#define TMF882X_MAX_MSG_SIZE
```

Value:

```
(64 + (TMF882X_HIST_NUM_TDC * \
                                        TMF882X_BYTES_PER_BIN * \
                                        TMF882X_HIST_NUM_BINS))
```

Maximum message size.

6.2.1.2 TOF_INIT_MSG

```
#define TOF_INIT_MSG(
              msg )
```

Value:

```
({ \
    struct tmf882x_msg *__m = (struct tmf882x_msg *)(msg); \
    memset(&__m->hdr, 0, TMF882X_MSG_HEADER_SIZE); \
})
```

6.2.1.3 TOF_SET_ERR_MSG

```
#define TOF_SET_ERR_MSG(
                msg,
                errid )
```

Value:

```
({ \
    struct tmf882x_msg *__m = (struct tmf882x_msg *)(msg); \
    TOF_SET_MSG_HDR(msg, ID_ERROR, struct tmf882x_msg_error); \
    __m->err_msg.err_code = errid; \
 })
```

6.2.1.4 TOF_SET_HISTOGRAM_MSG

```
#define TOF_SET_HISTOGRAM_MSG(
                msg,
                hist_type )
```

Value:

```
({ \
    struct tmf882x_msg *__m = (struct tmf882x_msg *)(msg); \
    TOF_SET_MSG_HDR(msg, ID_HISTOGRAM, struct tmf882x_msg_histogram); \
    __m->hist_msg.histogram_type = hist_type; \
 })
```

6.2.1.5 TOF_SET_MSG_HDR

```
#define TOF_SET_MSG_HDR(
                msg,
                id,
                type )
```

Value:

```
({ \
    struct tmf882x_msg *__m = (struct tmf882x_msg *)(msg); \
    TOF_INIT_MSG(msg); \
    __m->hdr.msg_id = id; \
    __m->hdr.msg_len = sizeof(type); \
})
```

6.2.1.6 TOF_ZERO_MSG

```
#define TOF_ZERO_MSG(
                msg )
```

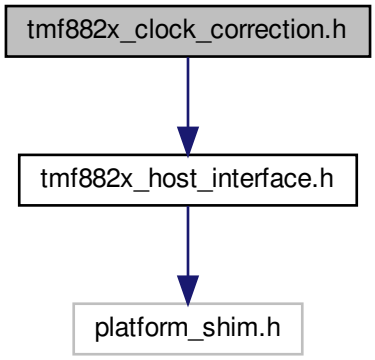
Value:

```
({ \
    struct tmf882x_msg *__m = (struct tmf882x_msg *)(msg); \
    memset(__m, 0, sizeof(struct tmf882x_msg)); \
})
```

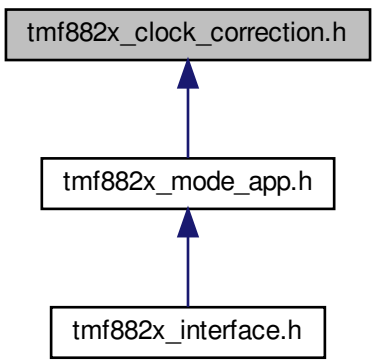
6.3 tmf882x_clock_correction.h File Reference

```
#include "tmf882x_host_interface.h"
```

Include dependency graph for tmf882x_clock_correction.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tmf882x_clk_corr

  This is the Context structure for the clock correction machine.

Functions

- void tmf882x_clk_corr_init (struct tmf882x_clk_corr ∗cr, uint32_t ratio)

  Initialize a clock correction context.
- void tmf882x_clk_corr_recalc (struct tmf882x_clk_corr ∗cr)

  Reset running clock correction state.
- void tmf882x_clk_corr_addpair (struct tmf882x_clk_corr ∗cr, uint32_t ref, uint32_t src)

  Add a pair of clock counts to the clock correction state.
- uint32_t tmf882x_clk_corr_map (struct tmf882x_clk_corr ∗cr, uint32_t old_val)

  Apply a clock correction mapping.

6.3.1 Function Documentation

6.3.1.1 tmf882x_clk_corr_addpair()

```
void tmf882x_clk_corr_addpair (
            struct tmf882x_clk_corr * cr,
            uint32_t ref,
            uint32_t src )
```

Add a pair of clock counts to the clock correction state.

Parameters

in	*cr*	pointer to clock correction context structure
in	*ref*	monotonic clock count for reference clock
in	*src*	monotonic clock count for source clock

Note

Which clock is used for reference and source does not matter as long as their use is consistent and is aligned with the expected ratio in tmf882x_clk_corr_init

6.3.1.2 tmf882x_clk_corr_init()

```
void tmf882x_clk_corr_init (
            struct tmf882x_clk_corr * cr,
            uint32_t ratio )
```

Initialize a clock correction context.

Parameters

in	*cr*	pointer to clock correction context structure
in	*ratio*	expected integer ratio of the two clocks

6.3.1.3 tmf882x_clk_corr_map()

```
uint32_t tmf882x_clk_corr_map (
            struct tmf882x_clk_corr * cr,
            uint32_t old_val )
```

Apply a clock correction mapping.

Parameters

in	*cr*	pointer to clock correction context structure
in	*old_val*	Value to be corrected by the clock correction mapping

Returns

Corrected value using the clock correction state. Default with no pairs added returns old_val.

6.3.1.4 tmf882x_clk_corr_recalc()

```
void tmf882x_clk_corr_recalc (
            struct tmf882x_clk_corr * cr )
```

Reset running clock correction state.

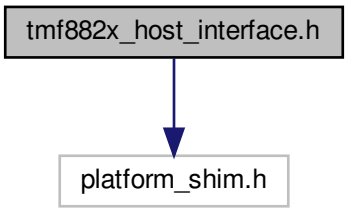
Parameters

in	*cr*	pointer to clock correction context structure

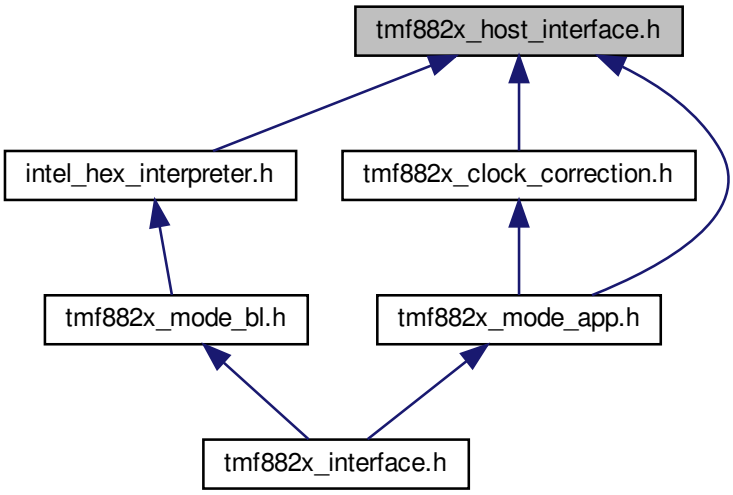
6.4 tmf882x_host_interface.h File Reference

```
#include "platform_shim.h"
```

Include dependency graph for tmf882x_host_interface.h:



This graph shows which files directly or indirectly include this file:



6.4.1 Detailed Description

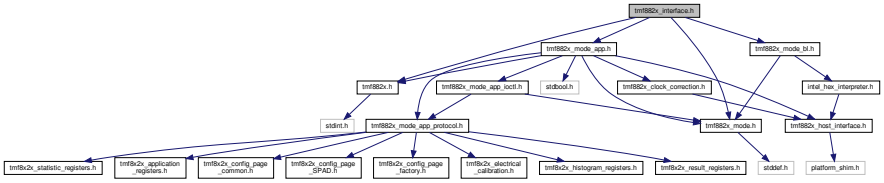
TMF882X host interface

6.5 tmf882x_interface.h File Reference

```
#include "tmf882x.h"
#include "tmf882x_mode.h"
#include "tmf882x_mode_bl.h"
```

```
#include "tmf882x_mode_app.h"
```

Include dependency graph for tmf882x_interface.h:



Classes

- struct tmf882x_tof

  TMF882X DCB context handle.

Macros

- #define **TMF882X_MAJ_MODULE_VER** 3
- #define **TMF882X_MIN_MODULE_VER** 7
- #define **QUOTE**(x) #x
- #define **STRINGIFY**(x) QUOTE(x)
- #define TMF882X_MODULE_VER

  TMF882X DCB driver module version string.
- #define TOF_FWDL_TIMEOUT_MSEC 30000

  Timeout for FWDL (firmware download)

Typedefs

- typedef enum tmf882x_mode_t **tmf882x_mode_t**
- typedef enum tmf882x_fwdl_type_t **tmf882x_fwdl_type_t**

Enumerations

- enum tmf882x_mode_t { TMF882X_MODE_BOOTLOADER = 0x80, TMF882X_MODE_APP = 0x03 }

  Supported application modes.
- enum tmf882x_fwdl_type_t { **FWDL_TYPE_BIN**, **FWDL_TYPE_HEX** }

  FWDL type when performing FWDL.
- enum tmf882x_regs {
  TMF882X_APP_ID = 0x00, TMF882X_STAT = 0xE0, TMF882X_INT_STAT = 0xE1, TMF882X_INT_EN = 0xE2,
  TMF882X_ID = 0xE3, TMF882X_REV_ID = 0xE4 }

  core register mapping common to all modes

Functions

- void tmf882x_init (struct tmf882x_tof *tof, void *priv)

  Initialize tof structure, must be called before using any other interface function.
- int32_t tmf882x_open (struct tmf882x_tof *tof)

  Open the firmware core driver interface. No effect if already open, error if a different application interface is already open.
- int32_t tmf882x_fwdl (struct tmf882x_tof *tof, tmf882x_fwdl_type_t fwdl_type, const uint8_t *buf, size_t len)

  Download new firmware. The new mode is automatically opened on success.
- int32_t tmf882x_mode_switch (struct tmf882x_tof *tof, tmf882x_mode_t mode)

  Perform an application mode switch operation on the current running application mode. The new mode is automatically opened on success.
- int32_t tmf882x_start (struct tmf882x_tof *tof)

  Start measurements with current configuration.
- int32_t tmf882x_process_irq (struct tmf882x_tof *tof)

  Check for interrupt conditions and handle accordingly. Any output data is passed through shim platform layer in the form of one or more tmf882x_msg.
- int32_t tmf882x_stop (struct tmf882x_tof *tof)

  Stop measurements.
- int32_t tmf882x_ioctl (struct tmf882x_tof *tof, uint32_t cmd, const void *input, void *output)

  Perform an IO Control command.
- void tmf882x_close (struct tmf882x_tof *tof)

  Close the core driver firmware interface, the inverse operation of tmf882x_open(). Device will be put into STANDBY power state. tmf882x_open() must be called again to perform any operations on the device.
- tmf882x_mode_t tmf882x_get_mode (struct tmf882x_tof *tof)

  Return the current mode.
- void tmf882x_set_debug (struct tmf882x_tof *tof, bool flag)

  Enable debug logging of the DCB.
- int32_t tmf882x_get_firmware_ver (struct tmf882x_tof *tof, char *ver, size_t len)

  Fill buffer with version string of the current open firmware mode.
- int32_t tmf882x_get_device_revision (struct tmf882x_tof *tof, char *rev_buf, size_t len)

  Fill buffer with device revision string.

6.5.1 Detailed Description

TMF882X Core Driver interface

6.5.2 Macro Definition Documentation

6.5.2.1 TMF882X_MODULE_VER

```
#define TMF882X_MODULE_VER
```

Value:

```
STRINGIFY(TMF882X_MAJ_MODULE_VER) "." \
                                  STRINGIFY(TMF882X_MIN_MODULE_VER)
```

TMF882X DCB driver module version string.

6.5.3 Enumeration Type Documentation

6.5.3.1 tmf882x_mode_t

```
enum tmf882x_mode_t
```

Supported application modes.

Enumerator

TMF882X_MODE_BOOTLOADER	Bootloader mode
TMF882X_MODE_APP	Application mode

6.5.3.2 tmf882x_regs

```
enum tmf882x_regs
```

core register mapping common to all modes

Enumerator

TMF882X_APP_ID	Application ID register
TMF882X_STAT	CPU Status register
TMF882X_INT_STAT	IRQ Status register
TMF882X_INT_EN	IRQ Enable register
TMF882X_ID	Chip ID register
TMF882X_REV_ID	Chip Revision register

6.5.4 Function Documentation

6.5.4.1 tmf882x_close()

```
void tmf882x_close (
            struct tmf882x_tof * tof )
```

Close the core driver firmware interface, the inverse operation of tmf882x_open(). Device will be put into STANDBY power state. tmf882x_open() must be called again to perform any operations on the device.

Parameters

in	*tof*	tof dcb interface context

Returns

0 for sucess, otherwise failure

6.5.4.2 tmf882x_fwdl()

```
int32_t tmf882x_fwdl (
            struct tmf882x_tof * tof,
            tmf882x_fwdl_type_t fwdl_type,
            const uint8_t * buf,
            size_t len )
```

Download new firmware. The new mode is automatically opened on success.

Parameters

in	*tof*	tof dcb interface context
in	*buf*	Firmware data buffer
in	*len*	size of firmware data buffer

Note

This function will try to re-open() the device after FWDL
This function supports partial Firmware Downloads when using intel hex record format. The return value will be negative and the device will not be re-opened until the EOF hex record is passed in.

Returns

0 for sucess, otherwise failure

6.5.4.3 tmf882x_get_device_revision()

```
int32_t tmf882x_get_device_revision (
            struct tmf882x_tof * tof,
            char * rev_buf,
            size_t len )
```

Fill buffer with device revision string.

Parameters

in	*tof*	pointer to tof dcb interface context
in	*rev_buf*	Buffer to be filled with version string
in	*len*	length of buffer

Returns

number of characters copied to buffer

6.5.4.4 tmf882x_get_firmware_ver()

```
int32_t tmf882x_get_firmware_ver (
            struct tmf882x_tof * tof,
            char * ver,
            size_t len )
```

Fill buffer with version string of the current open firmware mode.

Parameters

in	*tof*	pointer to tof dcb interface context
in	*ver*	Buffer to be filled with version string
in	*len*	length of buffer

Returns

number of characters copied to buffer, negative if an error occurred

6.5.4.5 tmf882x_get_mode()

```
tmf882x_mode_t tmf882x_get_mode (
            struct tmf882x_tof * tof )  [inline]
```

Return the current mode.

Parameters

in	*tof*	pointer to tof dcb interface context

Returns

current mode as tmf882x_mode_t

6.5.4.6 tmf882x_init()

```
void tmf882x_init (
            struct tmf882x_tof * tof,
            void * priv )
```

Initialize tof structure, *must be called* before using any other interface function.

Parameters

in	*tof*	pointer to tof dcb interface context to initialize
in	*priv*	Private context to pass back to client for callback platform functions

Warning

This function must be called before using any other interface function

Note

This function performs no I/O with the device

6.5.4.7 tmf882x_ioctl()

```
int32_t tmf882x_ioctl (
            struct tmf882x_tof * tof,
            uint32_t cmd,
            const void * input,
            void * output )
```

Perform an IO Control command.

Parameters

in	*tof*	Tof dcb interface context
in	*cmd*	Mode-dependent command code
in	*input*	An input argument passed to the ioctl command. See tmf882x_mode_app_ioctl.h for list of available ioctl commands and their respective argument(s). The type of input is dependent on the IOCTL command code. This argument may be NULL for commands that take no input.
out	*output*	An output argument returned from the ioctl command. See tmf882x_mode_app_ioctl.h for list of available ioctl commands and their respective argument(s). The type of output is dependent on the IOCTL command code. This argument may be NULL for commands that return no data.

Returns

0 for sucess, otherwise failure

6.5.4.8 tmf882x_mode_switch()

```
int32_t tmf882x_mode_switch (
            struct tmf882x_tof * tof,
            tmf882x_mode_t mode )
```

Perform an application mode switch operation on the current running application mode. The new mode is automatically opened on success.

Parameters

in	*tof*	tof dcb interface context
in	*mode*	Requested tmf882x_mode_t mode to switch to

Note

Not all modes may support switching to any other mode
This function will try to re-open() the device after mode switch

Returns

0 for sucess, otherwise failure

6.5.4.9 tmf882x_open()

```
int32_t tmf882x_open (
            struct tmf882x_tof * tof )
```

Open the firmware core driver interface. No effect if already open, error if a different application interface is already open.

Parameters

in	*tof*	tof dcb interface context

Warning

All other interface functions should not be called without calling tmf882x_open() first (except tmf882x_init).

Returns

0 for sucess, otherwise failure (driver remains closed on failure)

6.5.4.10 tmf882x_process_irq()

```
int32_t tmf882x_process_irq (
            struct tmf882x_tof * tof )  [inline]
```

Check for interrupt conditions and handle accordingly. Any output data is passed through shim platform layer in the form of one or more tmf882x_msg.

Parameters

in	*tof*	tof dcb interface context

Returns

0 for sucess, otherwise failure

6.5.4.11 tmf882x_set_debug()

```
void tmf882x_set_debug (
            struct tmf882x_tof * tof,
            bool flag )
```

Enable debug logging of the DCB.

Parameters

in	*tof*	pointer to tof dcb interface context
in	*flag*	Non-zero value enables debug logging, 0 disables debug logging

6.5.4.12 tmf882x_start()

```
int32_t tmf882x_start (
            struct tmf882x_tof * tof )  [inline]
```

Start measurements with current configuration.

Parameters

in	*tof*	tof dcb interface context

Returns

0 for sucess, otherwise failure

6.5.4.13 tmf882x_stop()

```
int32_t tmf882x_stop (
            struct tmf882x_tof * tof )  [inline]
```

Stop measurements.

Parameters

in	*tof*	tof dcb interface context

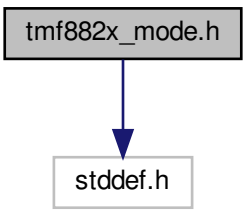
Returns

0 for sucess, otherwise failure

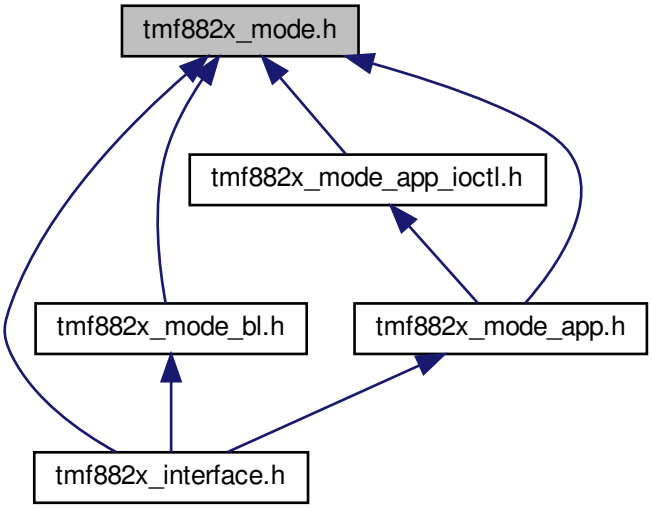
6.6 tmf882x_mode.h File Reference

```
#include <stddef.h>
```

Include dependency graph for tmf882x_mode.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct mode_vtable

  This is the Base mode behavioral function pointer structure.
- struct tmf882x_info_record

  This is the Base mode information record data.
- struct tmf882x_mode

  This is the Base mode context structure.

Macros

- #define MAX_REGS (256)

  max number of i2c registers
- #define member_of(ptr, type, member) ((type *) ((uint8_t *)(ptr) - offsetof(type, member)))

  macro to return pointer to parent struct from an embedded struct
- #define **_IOCTL_NRBITS** 8
- #define **_IOCTL_MODEBITS** 2
- #define **_IOCTL_ISIZEBITS** 10
- #define **_IOCTL_OSIZEBITS** 10
- #define **_IOCTL_DIRBITS** 2
- #define **_IOCTL_NRMASK** ((1 << _IOCTL_NRBITS)-1)
- #define **_IOCTL_MODEMASK** ((1 << _IOCTL_MODEBITS)-1)
- #define **_IOCTL_ISIZEMASK** ((1 << _IOCTL_ISIZEBITS)-1)
- #define **_IOCTL_OSIZEMASK** ((1 << _IOCTL_OSIZEBITS)-1)
- #define **_IOCTL_DIRMASK** ((1 << _IOCTL_DIRBITS)-1)
- #define **_IOCTL_NRSHIFT** 0
- #define **_IOCTL_MODESHIFT** (_IOCTL_NRSHIFT+_IOCTL_NRBITS)

- #define **_IOCTL_ISIZESHIFT** (_IOCTL_MODESHIFT+_IOCTL_MODEBITS)
- #define **_IOCTL_OSIZESHIFT** (_IOCTL_ISIZESHIFT+_IOCTL_ISIZEBITS)
- #define **_IOCTL_DIRSHIFT** (_IOCTL_OSIZESHIFT+_IOCTL_OSIZEBITS)
- #define **_IOCTL_READ** 2U
- #define **_IOCTL_WRITE** 1U
- #define **_IOCTL_NONE** 0U
- #define **_IOCTL**(mode, nr, dir, isize, osize)
- #define **_IOCTL_N**(mode, nr) _IOCTL((mode),(nr),_IOCTL_NONE,0,0)
- #define **_IOCTL_R**(mode, nr, otype) _IOCTL((mode),(nr),_IOCTL_READ,0,sizeof(otype))
- #define **_IOCTL_W**(mode, nr, itype) _IOCTL((mode),(nr),_IOCTL_WRITE,sizeof(itype),0)
- #define **_IOCTL_RW**(mode, nr, itype, otype) _IOCTL((mode),(nr),_IOCTL_READ|_IOCTL_WRITE,sizeof(itype),sizeof(otype))
- #define **_IOCTL_MODE**(nr) (((nr) >> _IOCTL_MODESHIFT) & _IOCTL_MODEMASK)
- #define **_IOCTL_DIR**(nr) (((nr) >> _IOCTL_DIRSHIFT) & _IOCTL_DIRMASK)
- #define **_IOCTL_NR**(nr) (((nr) >> _IOCTL_NRSHIFT) & _IOCTL_NRMASK)
- #define **_IOCTL_ISIZE**(nr) (((nr) >> _IOCTL_ISIZESHIFT) & _IOCTL_ISIZEMASK)
- #define **_IOCTL_OSIZE**(nr) (((nr) >> _IOCTL_OSIZESHIFT) & _IOCTL_OSIZEMASK)

Typedefs

- typedef enum tmf882x_pwr_mode_t **tmf882x_pwr_mode_t**

Enumerations

- enum tmf882x_pwr_mode_t { **TOF_STANDBY** = 0x00, **TOF_WAKEUP** = 0x01 }

  Indicate which power mode to switch to.

Functions

- void tmf882x_mode_init (struct tmf882x_mode *self, struct mode_vtable const *ops, void *priv)

  initialize a tmf882x_mode context structure
- void * tmf882x_mode_priv (struct tmf882x_mode *self)

  Return this mode's private context pointer.
- uint8_t tmf882x_mode (struct tmf882x_mode *self)

  Return this mode's information record mode ID.
- uint8_t tmf882x_mode_maj_ver (struct tmf882x_mode *self)

  Return this mode's major version number.
- int32_t tmf882x_mode_standby_operation (struct tmf882x_mode *self, tmf882x_pwr_mode_t mode)

  Configure a chip poweron/standby operation.
- int32_t tmf882x_mode_set_powerup_bootmatrix (struct tmf882x_mode *self, uint32_t powerup_bitfield)

  Set the powerup boot matrix of the device.
- int32_t tmf882x_mode_cpu_reset (struct tmf882x_mode *self, uint32_t powerup_bitfield)

  Perform a cpu reset.
- void tmf882x_mode_set_debug (struct tmf882x_mode *self, int32_t flag)

  Set debug logging for this mode.
- int32_t tmf882x_mode_version (struct tmf882x_mode *self, char *ver, size_t len)

  Fill buffer with mode version string.
- void tmf882x_dump_i2c_regs (struct tmf882x_mode *self)

  Log the i2c register map.
- void tmf882x_dump_data (struct tmf882x_mode *self, const uint8_t *buf, size_t len)

  Log the data buffer.

6.6.1 Detailed Description

TMF882X generic mode interface

6.6.2 Macro Definition Documentation

6.6.2.1 _IOCTL

```
#define _IOCTL(
               mode,
               nr,
               dir,
               isize,
               osize )
```

Value:

```
(((dir) << _IOCTL_DIRSHIFT) | \
      ((mode) << _IOCTL_MODESHIFT) | \
      ((nr)   << _IOCTL_NRSHIFT) | \
      ((isize) << _IOCTL_ISIZESHIFT) | \
      ((osize) << _IOCTL_OSIZESHIFT))
```

6.6.3 Function Documentation

6.6.3.1 tmf882x_dump_data()

```
void tmf882x_dump_data (
             struct tmf882x_mode * self,
             const uint8_t * buf,
             size_t len )
```

Log the data buffer.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*buf*	pointer to data buffer to log
in	*len*	length of data buffer

6.6.3.2 tmf882x_dump_i2c_regs()

```
void tmf882x_dump_i2c_regs (
            struct tmf882x_mode * self )
```

Log the i2c register map.

Parameters

in	*self*	pointer to tmf882x_mode context

6.6.3.3 tmf882x_mode()

```
uint8_t tmf882x_mode (
            struct tmf882x_mode * self )  [inline]
```

Return this mode's information record mode ID.

Parameters

in	*self*	pointer to tmf882x_mode context

Returns

mode ID value

6.6.3.4 tmf882x_mode_cpu_reset()

```
int32_t tmf882x_mode_cpu_reset (
            struct tmf882x_mode * self,
            uint32_t powerup_bitfield )
```

Perform a cpu reset.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*powerup_bitfield*	Bitfield to use for powerup matrix • 0x0: use default bootup • 0x1: Force boot monitor (bootloader) • 0x2: Force boot application currently in RAM

Returns

0 for success, otherwise failure

6.6.3.5 tmf882x_mode_init()

```
void tmf882x_mode_init (
            struct tmf882x_mode * self,
            struct mode_vtable const * ops,
            void * priv )
```

initialize a tmf882x_mode context structure

Parameters

in	*self*	pointer to tmf882x_mode context
in	*ops*	pointer to mode_vtable structure to set mode behavior
in	*priv*	User-private context to pass back through callback functions

6.6.3.6 tmf882x_mode_maj_ver()

```
uint8_t tmf882x_mode_maj_ver (
            struct tmf882x_mode * self )  [inline]
```

Return this mode's major version number.

Parameters

in	*self*	pointer to tmf882x_mode context

Returns

major version number

6.6.3.7 tmf882x_mode_priv()

```
void* tmf882x_mode_priv (
            struct tmf882x_mode * self )  [inline]
```

Return this mode's private context pointer.

Parameters

in	*self*	pointer to tmf882x_mode context

Returns

pointer to user-private context

6.6.3.8 tmf882x_mode_set_debug()

```
void tmf882x_mode_set_debug (
            struct tmf882x_mode * self,
            int32_t flag )  [inline]
```

Set debug logging for this mode.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*flag*	non zero flag value enables debug logging, 0 disables debug logging

6.6.3.9 tmf882x_mode_set_powerup_bootmatrix()

```
int32_t tmf882x_mode_set_powerup_bootmatrix (
            struct tmf882x_mode * self,
            uint32_t powerup_bitfield )
```

Set the powerup boot matrix of the device.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*powerup_bitfield*	Bitfield to use for powerup matrix • 0x0: use default bootup • 0x1: Force boot monitor (bootloader) • 0x2: Force boot application currently in RAM

Returns

0 for success, otherwise failure

6.6.3.10 tmf882x_mode_standby_operation()

```
int32_t tmf882x_mode_standby_operation (
            struct tmf882x_mode * self,
            tmf882x_pwr_mode_t mode )
```

Configure a chip poweron/standby operation.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*mode*	takes a tmf882x_pwr_mode_t to configure the power mode

Returns

0 for success, otherwise failure

6.6.3.11 tmf882x_mode_version()

```
int32_t tmf882x_mode_version (
            struct tmf882x_mode * self,
            char * ver,
            size_t len )
```

Fill buffer with mode version string.

Parameters

in	*self*	pointer to tmf882x_mode context
in	*ver*	Buffer to be filled with version string
in	*len*	length of buffer

Returns

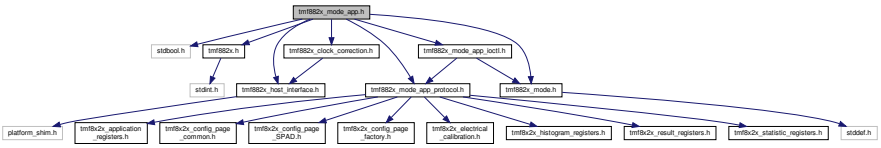
number of characters copied to buffer

6.7 tmf882x_mode_app.h File Reference

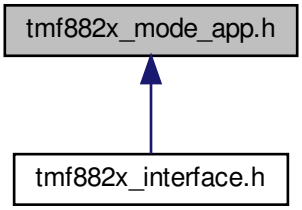
```
#include <stdbool.h>
#include "tmf882x.h"
#include "tmf882x_host_interface.h"
#include "tmf882x_mode_app_protocol.h"
#include "tmf882x_mode_app_ioctl.h"
#include "tmf882x_clock_correction.h"
```

```
#include "tmf882x_mode.h"
```

Include dependency graph for tmf882x_mode_app.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tmf882x_mode_app_i2c_msg

  App mode i2c message.
- struct tmf882x_mode_app

  This is the Application mode context structure.
- struct tmf882x_mode_app::volat_data

Macros

- #define mode_to_app(x)

  Return pointer to tmf882x_mode_app from pointer to tmf882x_mode.
- #define APP_MAX_MSG_SIZE

  Max i2c payload message size.

Enumerations

- enum tmf882x_mode_app_pckt_indices {
  APP_COM_RID_IDX = 0, **APP_COM_TID_IDX** = 1, **APP_COM_SIZE_LSB_IDX** = 2, APP_COM_SIZE_↩
  MSB_IDX = 3,
  APP_COM_MULTI_NUM_IDX = 4, **APP_COM_MULTI_SIZE_IDX** = 5, APP_COM_MULTI_CFG_ID_IDX =
  6, **APP_COM_DATA_IDX** = 4,
  APP_COM_MULTI_DATA_IDX = 7 }

  Indices for parsing i2c messages from the application.

Functions

- void tmf882x_mode_app_init (struct tmf882x_mode_app *app, void *priv)
  initialize a tmf882x_mode_app context structure

6.7.1 Detailed Description

TMF882X Application mode interface

6.7.2 Macro Definition Documentation

6.7.2.1 APP_MAX_MSG_SIZE

```
#define APP_MAX_MSG_SIZE
```

Value:

```
((TMF882X_HIST_NUM_BINS * \
                                TMF882X_HIST_NUM_TDC * \
                                TMF882X_BYTES_PER_BIN))
```

Max i2c payload message size.

6.7.2.2 mode_to_app

```
#define mode_to_app(
              x )
```

Value:

```
((struct tmf882x_mode_app *) \
     (member_of(x, struct tmf882x_mode_app, mode)))
```

Return pointer to tmf882x_mode_app from pointer to tmf882x_mode.

6.7.3 Enumeration Type Documentation

6.7.3.1 tmf882x_mode_app_pckt_indices

```
enum tmf882x_mode_app_pckt_indices
```

Indices for parsing i2c messages from the application.

Enumerator

APP_COM_SIZE_MSB_IDX	In multipacket msg 'DATA' is shifted down to make room for subpacket header
APP_COM_MULTI_CFG_ID_IDX	In multipacket msg 'DATA' is shifted down to make room for subpacket header

6.7.4 Function Documentation

6.7.4.1 tmf882x_mode_app_init()

```
void tmf882x_mode_app_init (
            struct tmf882x_mode_app * app,
            void * priv )
```

initialize a tmf882x_mode_app context structure

Parameters

in	*app*	pointer to app mode context
in	*priv*	User-private context to pass back through callback functions

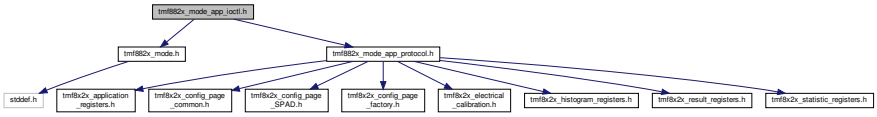
Note

Direct calls to this function should not be made, use the tmf882x_tof interface instead

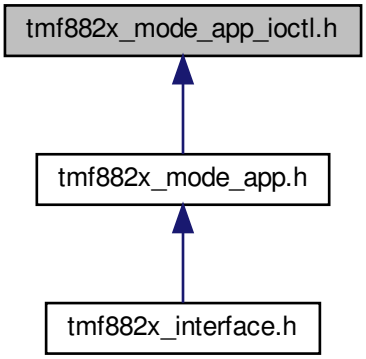
6.8 tmf882x_mode_app_ioctl.h File Reference

```
#include "tmf882x_mode.h"
#include "tmf882x_mode_app_protocol.h"
```

Include dependency graph for tmf882x_mode_app_ioctl.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tmf882x_mode_app_config

  This is the Application mode config structure that holds all configuration parameters for the application.
- struct tmf882x_mode_app_spad_config

  This is the Application mode spad config structure that holds the complete spad configuration for the application.
- struct tmf882x_mode_app_spad_config::tmf882x_mode_app_single_spad_config
- struct tmf882x_mode_app_calib

  This is the Application mode calibration structure.
- struct tmf882x_mode_app_dev_UID

  This is the Application mode structure to hold the device Unique ID.

Macros

- #define TMF882X_IOCTL_APP_MODE 0x02U

  APP mode IOCTL ID.
- #define **VERIFY_IOCAPP**(nr) (_IOCTL_MODE(nr) == TMF882X_IOCTL_APP_MODE)
- #define IOCAPP_SET_CFG _IOCTL_W(TMF882X_IOCTL_APP_MODE, APP_SET_CFG, struct tmf882x↩ _mode_app_config)

  IOCTL command code to Write a configuration to the application mode.
- #define IOCAPP_GET_CFG _IOCTL_R(TMF882X_IOCTL_APP_MODE, APP_GET_CFG, struct tmf882x↩ _mode_app_config)

  IOCTL command code to Read a configuration from the application mode.
- #define IOCAPP_SET_SPADCFG

  IOCTL command code to Write a spad configuration to the application mode.
- #define IOCAPP_GET_SPADCFG

  IOCTL command code to Read the spad configuration from the application mode.
- #define **TMF882X_MAX_CALIB_SIZE** (188 ∗ 4)
- #define IOCAPP_SET_CALIB

  IOCTL command code to Write the calibration data to the application mode.
- #define IOCAPP_GET_CALIB

IOCTL command code to Read the current calibration data from the application mode.

- #define IOCAPP_DO_FACCAL

  IOCTL command code to Perform Factory Calibration and get the new calibration data.

- #define IOCAPP_IS_MEAS

  IOCTL command code to Return whether the application mode is currently measuring.

- #define IOCAPP_DEV_UID

  IOCTL command code to Retrieve the device Unique Identifier (UID)

- #define IOCAPP_IS_CLKADJ

  IOCTL command code to Read the clock compensation enable state.

- #define IOCAPP_SET_CLKADJ

  IOCTL command code to Set the clock compensation enable state.

- #define IOCAPP_SET_8X8MODE

  IOCTL command code to Set the 8x8 operating mode (TMF8828)

- #define IOCAPP_IS_8X8MODE

  IOCTL command code to Read the 8x8 operating mode state (TMF8828)

Enumerations

- enum _tmf882x_mode_app_iocnr {
  APP_SET_CFG, **APP_GET_CFG**, **APP_SET_SPADCFG**, **APP_GET_SPADCFG**,
  APP_SET_CALIB, **APP_GET_CALIB**, **APP_DO_FACCAL**, **APP_IS_MEAS**,
  APP_DEV_UID, **APP_IS_CLKADJ**, **APP_SET_CLKADJ**, **APP_SET_8X8MODE**,
  APP_IS_8X8MODE, **NUM_APP_IOCTL** }

  These are the command code numberings in the ioctl bit fields. Do not use these directly, use the IOCTL bitmasks IOCAPP_∗.

6.8.1 Detailed Description

TMF882X APP mode ioctl definitions. See tmf882x_ioctl() for how to pass input and output parameters to IOCTL driver mode functions.

_IOCTL_R = IOCTL where driver returns data to the client through output param _IOCTL_W = IOCTL where client writes data to the driver through input param _IOCTL_RW = IOCTL data is both written by the client though the input param and data is returned through the output param _IOCTL_N = IOCTL where no data is written or returned

6.8.2 Macro Definition Documentation

6.8.2.1 IOCAPP_DEV_UID

```
#define IOCAPP_DEV_UID
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                    APP_DEV_UID, \
                                    struct tmf882x_mode_app_dev_UID )
```

IOCTL command code to Retrieve the device Unique Identifier (UID)

Parameters

in	*input*	type: none
out	*output*	type: struct tmf882x_mode_app_dev_UID *

Returns

zero for success, fail otherwise

6.8.2.2 IOCAPP_DO_FACCAL

```
#define IOCAPP_DO_FACCAL
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                    APP_DO_FACCAL, \
                                    struct tmf882x_mode_app_calib )
```

IOCTL command code to Perform Factory Calibration and get the new calibration data.

Parameters

in	*input*	type: none
out	*output*	type: struct tmf882x_mode_app_calib *

Returns

zero for success, fail otherwise

6.8.2.3 IOCAPP_GET_CALIB

```
#define IOCAPP_GET_CALIB
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                    APP_GET_CALIB, \
                                    struct tmf882x_mode_app_calib )
```

IOCTL command code to Read the current calibration data from the application mode.

Parameters

in	*input*	type: none
out	*output*	type: struct tmf882x_mode_app_calib *

Returns

zero for success, fail otherwise

6.8.2.4 IOCAPP_GET_CFG

```
#define IOCAPP_GET_CFG _IOCTL_R( TMF882X_IOCTL_APP_MODE, APP_GET_CFG, struct tmf882x_mode_↩
app_config )
```

IOCTL command code to Read a configuration from the application mode.

Parameters

in	*input*	type: none
out	*output*	type: struct tmf882x_mode_app_config *

Returns

zero for success, fail otherwise

6.8.2.5 IOCAPP_GET_SPADCFG

```
#define IOCAPP_GET_SPADCFG
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                    APP_GET_SPADCFG, \
                                    struct tmf882x_mode_app_spad_config )
```

IOCTL command code to Read the spad configuration from the application mode.

Parameters

in	*input*	type: none
out	*output*	type: struct tmf882x_mode_app_spad_config *

Returns

zero for success, fail otherwise

6.8.2.6 IOCAPP_IS_8X8MODE

```
#define IOCAPP_IS_8X8MODE
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                        APP_IS_8X8MODE, \
                                        bool )
```

IOCTL command code to Read the 8x8 operating mode state (TMF8828)

Parameters

in	*input*	type: none
out	*output*	type: bool *

Returns

zero for success, fail otherwise

6.8.2.7 IOCAPP_IS_CLKADJ

```
#define IOCAPP_IS_CLKADJ
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                        APP_IS_CLKADJ, \
                                        bool )
```

IOCTL command code to Read the clock compensation enable state.

Parameters

in	*input*	type: none
out	*output*	type: bool *

Returns

zero for success, fail otherwise

6.8.2.8 IOCAPP_IS_MEAS

```
#define IOCAPP_IS_MEAS
```

Value:

```
_IOCTL_R( TMF882X_IOCTL_APP_MODE, \
                                    APP_IS_MEAS, \
                                    bool )
```

IOCTL command code to Return whether the application mode is currently measuring.

Parameters

in	*input*	type: none
out	*output*	type: bool *

Returns

zero for success, fail otherwise

6.8.2.9 IOCAPP_SET_8X8MODE

```
#define IOCAPP_SET_8X8MODE
```

Value:

```
_IOCTL_W( TMF882X_IOCTL_APP_MODE, \
                                      APP_SET_8X8MODE, \
                                      bool )
```

IOCTL command code to Set the 8x8 operating mode (TMF8828)

Parameters

in	*input*	type: bool *
out	*output*	type: none

Returns

zero for success, fail otherwise Note that changing to/from 8x8 mode will reset the device configuration to its default.

6.8.2.10 IOCAPP_SET_CALIB

```
#define IOCAPP_SET_CALIB
```

Value:

```
_IOCTL_W( TMF882X_IOCTL_APP_MODE, \
                                        APP_SET_CALIB, \
                                        struct tmf882x_mode_app_calib )
```

IOCTL command code to Write the calibration data to the application mode.

Parameters

in	*input*	type: struct tmf882x_mode_app_calib *
out	*output*	type: none

Returns

zero for success, fail otherwise

6.8.2.11 IOCAPP_SET_CFG

```
#define IOCAPP_SET_CFG _IOCTL_W( TMF882X_IOCTL_APP_MODE, APP_SET_CFG, struct tmf882x_mode_↩
app_config )
```

IOCTL command code to Write a configuration to the application mode.

Parameters

in	*input*	type: struct tmf882x_mode_app_config *
out	*output*	type: none

Returns

zero for success, fail otherwise

6.8.2.12 IOCAPP_SET_CLKADJ

```
#define IOCAPP_SET_CLKADJ
```

Value:

```
_IOCTL_W( TMF882X_IOCTL_APP_MODE, \
                                        APP_SET_CLKADJ, \
                                        bool )
```

IOCTL command code to Set the clock compensation enable state.

Parameters

in	*input*	type: bool *
out	*output*	type: none

Returns

zero for success, fail otherwise

6.8.2.13 IOCAPP_SET_SPADCFG

```
#define IOCAPP_SET_SPADCFG
```

Value:

```
_IOCTL_W( TMF882X_IOCTL_APP_MODE, \
                                    APP_SET_SPADCFG, \
                                    struct tmf882x_mode_app_spad_config )
```

IOCTL command code to Write a spad configuration to the application mode.

Parameters

in	*input*	type: struct tmf882x_mode_app_spad_config *
out	*output*	type: none

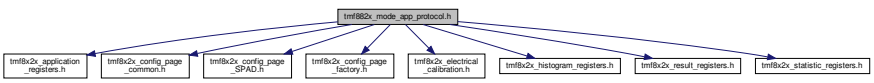
Returns

zero for success, fail otherwise

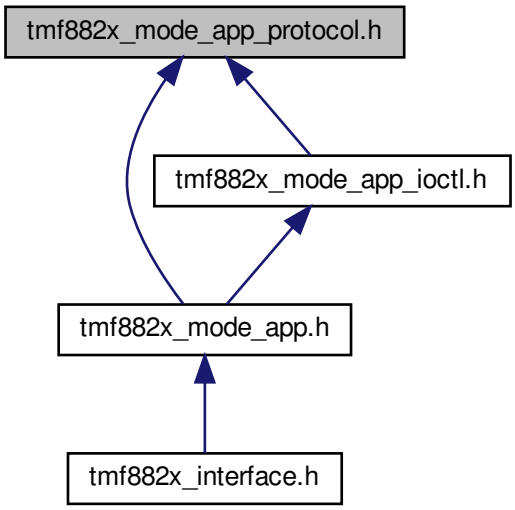
6.9 tmf882x_mode_app_protocol.h File Reference

```
#include "tmf8x2x_application_registers.h"
#include "tmf8x2x_config_page_common.h"
#include "tmf8x2x_config_page_SPAD.h"
#include "tmf8x2x_config_page_factory.h"
#include "tmf8x2x_electrical_calibration.h"
#include "tmf8x2x_histogram_registers.h"
#include "tmf8x2x_result_registers.h"
#include "tmf8x2x_statistic_registers.h"
```

Include dependency graph for tmf882x_mode_app_protocol.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **BITFIELD**(R, V) (((V)&((1UL<<R##__WIDTH)-1))<<R##__SHIFT)
- #define **BITMASKSHIFTED**(R) ((R##__MASK)>>(R##__SHIFT))
- #define **SET_BITFIELD**(R, D, V) (((D)&~((((1UL<<R##__WIDTH)-1)<<R##__SHIFT)))|(((V)&((1UL<<↩ R##__WIDTH)-1))<<R##__SHIFT))
- #define **GET_BITFIELD**(R, D) (((D)>>R##__SHIFT)&((1UL<<R##__WIDTH)-1))
- #define **TMF8X2X_BIT_MASK**(BF) (((1UL << (BF##__WIDTH)) - 1) << (BF##__SHIFT))
- #define **TMF8X2X_BIT_ONLY**(BF) (1UL << (BF##__SHIFT))
- #define TMF8X2X_COM_HEADER_SIZE ((TMF8X2X_COM_SIZE_MSB) - (TMF8X2X_COM_CONFIG_R↩ ESULT) + 1)
- #define **TMF8X2X_COM_HEADER_PLUS_PAYLOAD** ((0xDF) - (TMF8X2X_COM_CONFIG_RESULT) + 1)
- #define **TMF8X2X_COM_MAX_PAYLOAD** ((TMF8X2X_COM_HEADER_PLUS_PAYLOAD) - (TMF8X2↩ X_COM_HEADER_SIZE))
- #define TMF8X2X_COM_OPTIONAL_SUBPACKET_HEADER_SIZE 3 /** if there is a sub-packet, this is the size of the sub-packet header */
- #define **TMF8X2X_COM_OPTIONAL_SUBPACKET_HEADER_MASK** (0x80) /** this is the bit that has to be set to indicated in the RID that there is a sub-packet header */
- #define **TMF8X2X_COM_OPTIONAL_SUBPACKET_NUMBER_ADDRESS** ((TMF8X2X_COM_SIZE_L↩ SB)+2) /** the sub-packet header is right after the packet header */
- #define **TMF8X2X_COM_OPTIONAL_SUBPACKET_PAYLOAD_ADDRESS** ((TMF8X2X_COM_OPTIO↩ NAL_SUBPACKET_NUMBER_ADDRESS)+1) /** the size of the payload in the sub-packet comes as 2nd byte of the sub-packet header */
- #define **TMF8X2X_COM_OPTIONAL_SUBPACKET_CONFIG_ID_ADDRESS** ((TMF8X2X_COM_OPTI↩ ONAL_SUBPACKET_NUMBER_ADDRESS)+2) /** the CONFIG ID is the subcapture number of this sub-packet message or breakpoint #*/
- #define TMF8X2X_COM_RID_FOR_HISTOGRAM(histType) ((TMF8X2X_COM_OPTIONAL_SUBPACK↩ ET_HEADER_MASK) | (histType))

- #define TMF8X2X_COM_RID_RAW_HISTOGRAM_24_BITS (TMF8X2X_COM_RID_FOR_HISTOGRAM(TMF8X2X_COM_HIST_DUMP__histogram__raw_24_bit_histogram))
- #define **TMF8X2X_COM_RID_ELECTRICAL_CALIBRATION_24_BITS** (TMF8X2X_COM_RID_FOR_HISTOGRAM(TMF8X2X_COM_HIST_DUMP__histogram__electrical_calibration_24_bit_histogram))
- #define **TMF8X2X_COM_RID_BREAKPOINT_HIT** (TMF8X2X_COM_RID_FOR_HISTOGRAM(TMF8X2X_COM_HIST_DUMP__histogram__breakpoint_hit)) /** Breakpoint information - generic format */
- #define **TMF8X2X_COM_MAX_MEASUREMENT_RESULTS** (36)
- #define **TMF8X2X_COM_MAX_SPAD_XSIZE** (18)
- #define **TMF8X2X_COM_MAX_SPAD_YSIZE** (10)
- #define **TMF8X2X_COM_MAX_SPAD_SIZE**
- #define **TMF8X2X_MAX_CONFIGURATIONS** 2
- #define **TMF8X2X_MAIN_SPAD_VERTICAL_LSB_SHIFT** (0)
- #define **TMF8X2X_MAIN_SPAD_VERTICAL_MID_SHIFT** (10)
- #define **TMF8X2X_MAIN_SPAD_VERTICAL_MSB_SHIFT** (20)
- #define TMF8X2X_MAIN_SPAD_BITS_PER_CHANNEL (3)
- #define TMF8X2X_MAIN_SPAD_ENCODE_CHANNEL(channel, yPosition)
- #define TMF8X2X_MAIN_SPAD_DECODE_CHANNEL(config, yPosition)

6.9.1 Macro Definition Documentation

6.9.1.1 TMF8X2X_COM_HEADER_SIZE

```
#define TMF8X2X_COM_HEADER_SIZE ((TMF8X2X_COM_SIZE_MSB) - (TMF8X2X_COM_CONFIG_RESULT) + 1)
```

Protocol Structure for results/readout are: If the size fits into one packet (i.e. is less than 0xC0, than there is no payload field).

+—+—+—+—+—+—+-...-+----—+

| RID | TID | Size | Size | Data0 | Data1 | ... | Data[Size-1] |

| a | | LSB | MSB | | | | |

+—+—+—+—+—+—+-...-+----—+

If the data to be transferred does not fit in one I2C Chunk, i.e. is Bigger than (0xE0 - 0x20)=0xC0, than it is split into the following records (each record starts at address TMF8X2C_COM_RESULT_ADDRESS:

+—+—+—+—+———+———+———+-...-+—-—+

| RID | TID | Total Size | Payload | Data[0] | Data[1] | ... | Data[m-1] |

| b | n | LSB | MSB | m | | | | |

+—+—+—+—+———+———+———+-...-+—-—+

+—+—+—+—+———+———+—-—+-...-+--------—+

| RID | RID | Total Size | Payload | Data[m] | Data[m+1] | ... | Data[m + (m+1)] |

| b | n+1 | - Payload m | m+1 | | | | |

+—+—+—+—+———+———+—-—+-...-+--------—+

....

Note that the TID changes with each new packet, the Size is reduced with each new packet, the payload may change with each packet (most likely it stays the same - and only changes with the last packet - if at all), and the RID always stays the same.TID (1Byte), RID (1Byte), Size (2Bytes) every response has this header (4 bytes)

6.9.1.2 TMF8X2X_COM_MAX_SPAD_SIZE

```
#define TMF8X2X_COM_MAX_SPAD_SIZE
```

Value:

```
(TMF8X2X_COM_MAX_SPAD_XSIZE * \
                                        TMF8X2X_COM_MAX_SPAD_YSIZE)
```

6.9.1.3 TMF8X2X_COM_OPTIONAL_SUBPACKET_HEADER_SIZE

```
#define TMF8X2X_COM_OPTIONAL_SUBPACKET_HEADER_SIZE 3 /** if there is a sub-packet, this is the
size of the sub-packet header */
```

If the RID has the TMF8X2X_COM_SUB_PACKET_HEADER_MASK bit set, than there is a subheader immediatly after the header.

The optional sub-packet header consists of 2 bytes: A running number identifying which chunk of the complete packet is transitted. A payload byte giving the amount of data that is following in this sub-packet.

Note: The total size written to TMF8X2X_COM_SIZE_LSB *excludes* the sub-packet headers! This makes it easier, otherwise we would need to calculate into how many sub-packets we cut the complete packet before starting to send.

6.9.1.4 TMF8X2X_COM_RID_FOR_HISTOGRAM

```
#define TMF8X2X_COM_RID_FOR_HISTOGRAM(
             histType ) ( (TMF8X2X_COM_OPTIONAL_SUBPACKET_HEADER_MASK) | (histType) )
```

Histogram types build the RID for histograms from select bit + sub-packet header mask: histograms never fit in 1 I2C packet

6.9.1.5 TMF8X2X_COM_RID_RAW_HISTOGRAM_24_BITS

```
#define TMF8X2X_COM_RID_RAW_HISTOGRAM_24_BITS ( TMF8X2X_COM_RID_FOR_HISTOGRAM( TMF8X2X_COM_H↩
IST_DUMP__histogram__raw_24_bit_histogram ) )
```

The combination of the original RID + sub-packet indication

6.9.1.6 TMF8X2X_MAIN_SPAD_BITS_PER_CHANNEL

```
#define TMF8X2X_MAIN_SPAD_BITS_PER_CHANNEL ( 3 )
```

each channel can be encoded in 3 bits

6.9.1.7 TMF8X2X_MAIN_SPAD_DECODE_CHANNEL

```
#define TMF8X2X_MAIN_SPAD_DECODE_CHANNEL(
              config,
              yPosition )
```

Value:

```
( ( ( ( (config) >> ( (TMF8X2X_MAIN_SPAD_VERTICAL_LSB_SHIFT) + (yPosition) ) )      ) & 1 /*LSB*/ )
        \
        | ( ( ( (config) >> ( (TMF8X2X_MAIN_SPAD_VERTICAL_MID_SHIFT) + (yPosition) ) ) << 1 ) & 2 /*Mid*/ )
                \
        | ( ( ( (config) >> ( (TMF8X2X_MAIN_SPAD_VERTICAL_MSB_SHIFT) + (yPosition) ) ) << 2 ) & 4 /*MSB*/ )
                \
        )
```

to decode a single channel, we need to get 3 bits at specific positions in the 32-bit word

6.9.1.8 TMF8X2X_MAIN_SPAD_ENCODE_CHANNEL

```
#define TMF8X2X_MAIN_SPAD_ENCODE_CHANNEL(
              channel,
              yPosition )
```

Value:

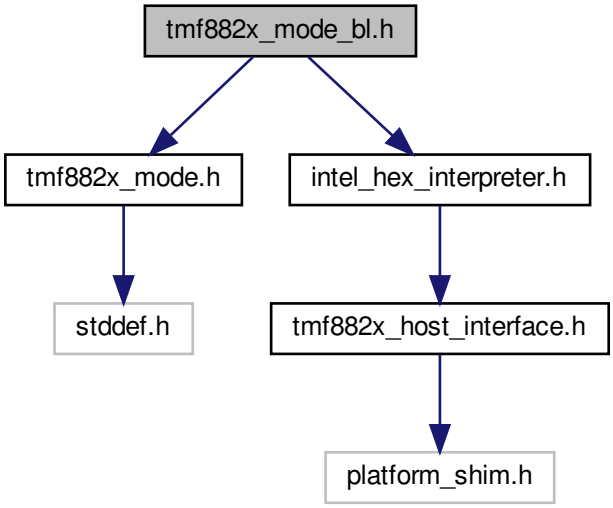
```
( ( ( ( (channel) & 1 /*LSB*/ )      ) << ( (TMF8X2X_MAIN_SPAD_VERTICAL_LSB_SHIFT) + (yPosition) ) )
        \
        | ( ( ( (channel) & 2 /*Mid*/ ) >> 1 ) << ( (TMF8X2X_MAIN_SPAD_VERTICAL_MID_SHIFT) + (yPosition) )
      )         \
        | ( ( ( (channel) & 4 /*MSB*/ ) >> 2 ) << ( (TMF8X2X_MAIN_SPAD_VERTICAL_MSB_SHIFT) + (yPosition) )
      )         \
        )
```

to encode a single channel, we need to set 3 bits at specific positions in the 32-bit word

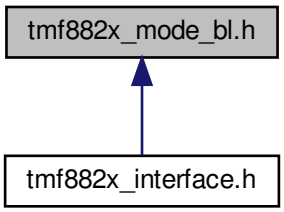
6.10 tmf882x_mode_bl.h File Reference

```
#include "tmf882x_mode.h"
#include "intel_hex_interpreter.h"
```

Include dependency graph for tmf882x_mode_bl.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct tmf882x_mode_bl_short_resp
- struct tmf882x_mode_bl_read_ram_resp
- struct tmf882x_anon_resp
- union tmf882x_mode_bl_response
- struct tmf882x_mode_bl_short_cmd
- struct tmf882x_mode_bl_upload_init_cmd
- struct tmf882x_mode_bl_read_ram_cmd
- struct tmf882x_mode_bl_write_ram_cmd
- struct tmf882x_mode_bl_addr_ram_cmd
- struct tmf882x_mode_bl_anon_cmd
- union tmf882x_mode_bl_command
- struct tmf882x_mode_bl

Macros

- #define mode_to_bl(x)

  Return pointer to tmf882x_mode_bl from pointer to tmf882x_mode.
- #define **BL_CMD_SIZE** 1
- #define **BL_DATA_LEN_SIZE** 1
- #define **BL_CHKSUM_SIZE** 1
- #define **BL_NUM_DATA** 128
- #define **BL_MAX_DATA_SZ** (BL_NUM_DATA∗sizeof(uint8_t))
- #define **BL_MSG_HEADER_SIZE**
- #define **BL_MSG_FOOTER_SIZE** BL_CHKSUM_SIZE
- #define **BL_MSG_CMD_MAX_SIZE**
- #define **BL_CALC_CHKSUM_SIZE**(sz)
- #define **BL_CALC_CMD_SIZE**(sz)
- #define **BL_CALC_RSP_SIZE**(sz)
- #define BL_IS_CMD_BUSY(x) ((x) >= BL_STAT_CMD_BUSY)

  Returns non-zero if the CMD_STAT value indicates the bootloader is busy performing a command.

Enumerations

- enum tmf882x_mode_bl_regs {
  BL_REG_CMD_STATUS = 0x08, **BL_REG_DATA_SIZE** = 0x09, **BL_REG_DATA_0** = 0x0A, **BL_REG_↩**
  DATA_127 = 0x89,
  BL_REG_CHKSUM = 0x8A }

  register map specific to bootloader mode
- enum tmf882x_mode_bl_cmd {
  BL_CMD_RST = 0x10, **BL_CMD_RAMREMAP_RST** = 0x11, **BL_CMD_ROMREMAP_RST** = 0x12, **BL_↩**
  CMD_EXT_FLASHRST = 0x13,
  BL_CMD_UPLOAD_INIT = 0x14, **BL_CMD_ADDR_EXT_FLASH** = 0x15, **BL_CMD_BIST** = 0x2C, **BL_C↩**
  MD_RD_RAM = 0x40,
  BL_CMD_WR_RAM = 0x41, **BL_CMD_RAM_ADDR** = 0x43 }

  all bootloader mode commands
- enum tmf882x_mode_bl_cmd_stat {
  BL_STAT_READY = 0x0, **BL_STAT_ERR_SIZE** = 0x1, **BL_STAT_ERR_CSUM** = 0x2, **BL_STAT_ERR_↩**
  RES = 0x3,
  BL_STAT_ERR_APP = 0x4, **BL_STAT_ERR_TIMEOUT** = 0x5, **BL_STAT_ERR_LOCK** = 0x6, **BL_STA↩**
  T_ERR_RANGE = 0x7,
  BL_STAT_ERR_MORE = 0x8, **BL_STAT_ERROR1** = 0x9, **BL_STAT_ERROR2** = 0xA, **BL_STAT_ERR↩**
  OR3 = 0xB,
  BL_STAT_ERROR4 = 0xC, **BL_STAT_ERROR5** = 0xD, **BL_STAT_ERROR6** = 0xE, **BL_STAT_ERROR7** =
  0xF,
  BL_STAT_CMD_BUSY = 0x10, **MAX_BL_STAT** }

  all bootloader mode command status return codes

Functions

- void tmf882x_mode_bl_init (struct tmf882x_mode_bl ∗bl, void ∗priv)

6.10.1 Detailed Description

TMF882X Bootloader mode interface

6.10.2 Macro Definition Documentation

6.10.2.1 BL_CALC_CHKSUM_SIZE

```
#define BL_CALC_CHKSUM_SIZE(
              sz )
```

Value:

```
((sz) + \
                                  BL_MSG_HEADER_SIZE)
```

6.10.2.2 BL_CALC_CMD_SIZE

```
#define BL_CALC_CMD_SIZE(
              sz )
```

Value:

```
(BL_CALC_CHKSUM_SIZE(sz) + \
                                  BL_MSG_FOOTER_SIZE)
```

6.10.2.3 BL_CALC_RSP_SIZE

```
#define BL_CALC_RSP_SIZE(
              sz )
```

Value:

```
((sz) + \
                                  BL_MSG_HEADER_SIZE + \
                                  BL_MSG_FOOTER_SIZE)
```

6.10.2.4 BL_MSG_CMD_MAX_SIZE

```
#define BL_MSG_CMD_MAX_SIZE
```

Value:

```
(BL_MSG_HEADER_SIZE   + \
                                  BL_MAX_DATA_SZ + \
                                  BL_MSG_FOOTER_SIZE)
```

6.10.2.5 BL_MSG_HEADER_SIZE

```
#define BL_MSG_HEADER_SIZE
```

Value:

```
(BL_CMD_SIZE + \
                                    BL_DATA_LEN_SIZE)
```

6.10.2.6 mode_to_bl

```
#define mode_to_bl(
               x )
```

Value:

```
((struct tmf882x_mode_bl *) \
     (member_of(x, struct tmf882x_mode_bl, mode)))
```

Return pointer to tmf882x_mode_bl from pointer to tmf882x_mode.

6.10.3 Function Documentation

6.10.3.1 tmf882x_mode_bl_init()

```
void tmf882x_mode_bl_init (
             struct tmf882x_mode_bl * bl,
             void * priv )
```

Initialize a tmf882x_mode_bl context structure

Parameters

in	*bl*	pointer to bl mode context
in	*priv*	User-private context to pass back through callback functions

Note

Direct calls to this function should not be made, use the tmf882x_tof interface instead