|  |
| --- |
| ADI VSM Study Watch  Driver Reference Guide |
| , |

Table of Contents

[1 Introduction 5](#_Toc64359922)

[1.1 Scope 5](#_Toc64359923)

[1.2 Organization of this Guide 5](#_Toc64359924)

[1.3 Acronyms 5](#_Toc64359925)

[1.4 References 5](#_Toc64359926)

[1.5 Additional Information 5](#_Toc64359927)

[2 Specifications 6](#_Toc64359928)

[2.1 Version Information 6](#_Toc64359929)

[3 ADPD4100 API Guide 7](#_Toc64359930)

[3.1 API Functions 7](#_Toc64359931)

[3.1.1 Adpd400xDrvOpenDriver 7](#_Toc64359932)

[3.1.2 Adpd400xDrvCloseDriver 7](#_Toc64359933)

[3.1.3 Adpd400xDrvGetComMode 8](#_Toc64359934)

[3.1.4 Adpd400xDrvRegWrite 8](#_Toc64359935)

[3.1.5 Adpd400xDrvRegRead 8](#_Toc64359936)

[3.1.6 Adpd400xDrvRegRead32B 9](#_Toc64359937)

[3.1.7 Adpd400xDrvSetOperationMode 10](#_Toc64359938)

[3.1.8 Adpd400xDrvSetOperationPause 10](#_Toc64359939)

[3.1.9 Adpd400xDrvSlotSetup 10](#_Toc64359940)

[3.1.10 Adpd400xDrvSlotSetActive 11](#_Toc64359941)

[3.1.11 Adpd400xDrvDataReadyCallback 12](#_Toc64359942)

[3.1.12 Adpd400xISR 12](#_Toc64359943)

[3.1.13 Adpd400xDrvSetParameter 12](#_Toc64359944)

[3.1.14 Adpd400xDrvGetParameter 13](#_Toc64359945)

[3.1.15 Adpd400xDrvReadFifoData 14](#_Toc64359946)

[3.1.16 Adpd400xDrvReadRegData 15](#_Toc64359947)

[3.1.17 Adpd400xDrvSetLedCurrent 15](#_Toc64359948)

[3.1.18 Adpd400xDrvGetLedCurrent 16](#_Toc64359949)

[3.1.19 Adpd400xDrvSoftReset 17](#_Toc64359950)

[3.1.20 \_Adpd400xDrvInit 17](#_Toc64359951)

[3.1.21 \_Adpd400xDrvSetInterrupt 17](#_Toc64359952)

[3.1.22 \_Adpd400xDrvSetIdleMode 18](#_Toc64359953)

[3.1.23 \_Adpd400xDrvGetSlotInfo 18](#_Toc64359954)

[3.1.24 \_Adpd400xDrvGetDataOuputRate 18](#_Toc64359955)

[3.1.25 \_Adpd400xDrvSlotSaveCurrentSetting 19](#_Toc64359956)

[3.1.26 \_Adpd400xDrvSlotApplyPreviousSetting 19](#_Toc64359957)

[3.1.27 \_Adpd400xDrvSlotApplySkipSetting 20](#_Toc64359958)

[3.1.28 \_Adpd400xDrvSetSlotSize 20](#_Toc64359959)

[3.1.29 \_FifoLevel 20](#_Toc64359960)

[3.2 API Data Types 21](#_Toc64359961)

[3.2.1 adpd400xDrv\_slot\_t 21](#_Toc64359962)

[3.3 Enums 22](#_Toc64359963)

[3.3.1 ADPD400xDrv\_Operation\_Mode\_t 22](#_Toc64359964)

[3.3.2 ADPD400XDrv\_FIFO\_SIZE\_t 22](#_Toc64359965)

[3.3.3 Adpd400xCommandStruct\_t 22](#_Toc64359966)

[3.3.4 ADPDDrvCl\_SignalDark\_t 23](#_Toc64359967)

[3.3.5 ADPD400xDrv\_SlotNum\_t 23](#_Toc64359968)

[3.3.6 ADPD400xDrv\_LedId\_t 23](#_Toc64359969)

[3.3.7 Adpd400xComMode\_t 24](#_Toc64359970)

List of Figures

No table of figures entries found.

List of Tables

No table of figures entries found.Copyright, Disclaimer & Trademark Statements

Copyright Information

Copyright (c) 2021 Analog Devices, Inc. All Rights Reserved. This documentation is proprietary and confidential to Analog Devices, Inc. and its licensors. This document may not be reproduced in any form without prior, express consent from Analog Devices, Inc.

Disclaimer

Analog Devices, Inc. (“Analog Devices”) reserves the right to change this product without prior notice. Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use; nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under the patent or other rights of Analog Devices

Trademark and Service Mark Notice

Analog Devices, the Analog Devices logo, Blackfin, SHARC, TigerSHARC, CrossCore, VisualDSP, VisualDSP++, EZ-KIT Lite, EZ-Extender, SigmaStudio and Collaborative are the exclusive trademarks and/or registered trademarks of Analog Devices, Inc (“Analog Devices”).

All other brand and product names are trademarks or service marks of their respective owners.

Analog Devices’ Trademarks and Service Marks may not be used without the express written consent of Analog Devices, such consent only to be provided in a separate written agreement signed by Analog Devices. Subject to the foregoing, such Trademarks and Service Marks must be used according to Analog Devices’ Trademark Usage guidelines. Any licensee wishing to use Analog Devices’ Trademarks and Service Marks must obtain and follow these guidelines for the specific marks at issue.

# Introduction

This is a living document describing the various device drivers used on ADI VSM Study Watch.

## Scope

The API of the drivers, to enable software developers to use in a C application, and for system integrators to build it into a complete system. Developers are assumed to be familiar with the ADI sensors/accelerometer.

## Organization of this Guide

Section [1:](#_Introduction) this section contains the introduction

Section [2:](#_Specifications_1) lists specifications of the devices

Section [3:](#_ADPD4100_API_Guide) ADPD4100 driver API guide

Section [4:](#chk1) ADXL362 driver API guide

## Acronyms

|  |  |
| --- | --- |
| **ADI** | **A**nalog **D**evices **I**nc. |
| **API** | **A**pplication **P**rogram **I**nterface |
| **ADPD** | **A**nalog **D**evices **P**hoto **D**iode Sensor |

## References

1. [ADPD4100](https://www.analog.com/media/en/technical-documentation/data-sheets/adpd4100-4101.pdf) Data sheet
2. [ADXL362](https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL362.pdf) Data sheet

## Additional Information

For more information on the latest ADI processors, sensors, silicon errata, code examples,

development tools, system services and devices drivers, technical support and any other

additional information, please email healthcare-support@analog.com, or visit our website at www.analog.com/processors.

# Specifications

The drivers covered in this document include

* ADPD4100 – This is the latest multimodal sensor front end, which can stimulate up to eight LEDs and measuring the return signal on up to eight separate current inputs. It has twelve time slots which enables 12 separate measurements per sampling period. The driver utilizes an SPI interface for data output and functional configuration.

* ADXL362 – This is the ultralow power, 3-axis accelerometer which provides 12-bit output resolution supporting measurement ranges of +/-2g, +/-4g and +/-8g.

## Version Information

This document describes release 1.0.0 of the ADI VSM Study Watch device driver document.

# ADPD4100 API Guide

The driver comes with the following header files:

* adpd400x\_drv.h – This includes the API definitions, macros, enums and structures used in driver.
* adpd400x\_reg.h – This file contains the register addresses and bit definitions of each register.

## API Functions

### Adpd400xDrvOpenDriver

**Prototype**

int16\_t Adpd400xDrvOpenDriver(void;

**Description**

This function sets up the interface lines, initialization of driver and interrupt mode as FIFO. It can check whether the interface is SPI or I2c and accordingly set the communication mode.

**Parameters**

None.

**Return value**

ADPD400xDrv\_SUCCESS – Driver open successful

ADPD400xDrv\_ERROR – Error in opening the driver

### Adpd400xDrvCloseDriver

**Prototype**

int16\_t Adpd400xDrvCloseDriver(void);

**Description**

Sets up the device in idle mode and closes the driver.

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Driver close successful

ADPD400xDrv\_ERROR – Error in closing the driver

### Adpd400xDrvGetComMode

**Prototype**

Adpd400xComMode\_t Adpd400xDrvGetComMode();

**Description**

Returns the communication bus: I2C, SPI or unknown

**Parameters**

None

**Return value**

Adpd400xComMode\_t: ADPD400x\_I2C\_BUS, ADPD400x\_SPI\_BUS or ADPD400x\_UNKNOWN\_BUS

### Adpd400xDrvRegWrite

**Prototype**

int16\_t Adpd400xDrvRegWrite(uint16\_t nAddr, uint16\_t nRegValue);

**Description**

This function does the synchronous register write of a 16-bit value to the device

**Parameters**

**Name:** nAddr

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Register address to write

**Name:** nRegValue

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Value to write to register

**Return value**

ADPD400xDrv\_SUCCESS – Write to register successful

ADPD400xDrv\_ERROR – Error in writing to register

### Adpd400xDrvRegRead

**Prototype**

int16\_t Adpd400xDrvRegRead(uint16\_t nAddr, uint16\_t \*pnData);

**Description**

This function does the synchronous register read of a 16-bit value from the device

**Parameters**

**Name:** nAddr

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Register address to read

**Name:** pnData

**Type:** uint16\_t \*

**Direction:** Input/Output

**Description:** Pointer location to read the value into from register

**Return value**

ADPD400xDrv\_SUCCESS – Read from register successful

ADPD400xDrv\_ERROR – Error in reading register

### Adpd400xDrvRegRead32B

**Prototype**

int16\_t Adpd400xDrvRegRead32B(uint16\_t nAddr, uint32\_t \*pnData);

**Description**

This function does the synchronous register read of a 32-bit value from the device

**Parameters**

**Name:** nAddr

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Register address to read

**Name:** pnData

**Type:** uint32\_t \*

**Direction:** Input/Output

**Description:** Pointer location to read the value into from register

**Return value**

ADPD400xDrv\_SUCCESS – Read from register successful

ADPD400xDrv\_ERROR – Error in reading register

### Adpd400xDrvSetOperationMode

**Prototype**

int16\_t Adpd400xDrvSetOperationMode(uint8\_t nOpMode);

**Description**

This function sets the operating mode of the device. And accordingly clears the FIFO.

**Parameters**

**Name:** nOpMode

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Mode to set

**Return value**

ADPD400xDrv\_SUCCESS – Setting mode successful

ADPD400xDrv\_ERROR – Error in setting mode

### Adpd400xDrvSetOperationPause

**Prototype**

int16\_t Adpd400xDrvSetOperationMode(uint8\_t nEnable);

**Description**

This function sets the operating mode of the device to pause mode. And accordingly clears the FIFO.

**Parameters**

**Name:** nOpMode

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Mode to set

**Return value**

ADPD400xDrv\_SUCCESS – Setting pause mode successful

ADPD400xDrv\_ERROR – Error in setting pause mode

### Adpd400xDrvSlotSetup

**Prototype**

int16\_t Adpd400xDrvSlotSetup(uint8\_t nSlotNum, uint8\_t nEnable, uint16\_t nSlotFormat, uint8\_t nChannel);

**Description**

Sets up the slot for operation. The slot number starts from 0 for Slot A and ends with 11 for Slot L.

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot number to set

**Name:** eEnable

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** If this field is 0, then it disables all slots after the *nSlotNum*. If this field is 1, then it sets all slots

before the *nSlotNum*.

**Name:** nSlotFormat

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Slot format for setting the data format in FIFO (Impulse, Dark, Sig)

**Name:** nChannel

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Number of channels to enable

**Return value**

ADPD400xDrv\_SUCCESS – Setting slot successful

ADPD400xDrv\_ERROR – Error in setting slot

### Adpd400xDrvSlotSetActive

**Prototype**

int16\_t Adpd400xDrvSlotSetActive(uint8\_t nSlotNum, uint8\_t nActive);

**Description**

Sets up a slot in sleep or active mode. The slot number starts from 0 for Slot A and ends with 11 for Slot L.

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot number to set

**Name:** nActive

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** If this field is 0, then it puts the slot into sleep. If this field is 1, then it sets the slot in awake

mode.

**Return value**

ADPD400xDrv\_SUCCESS – Setting slot into sleep/active mode successful

ADPD400xDrv\_ERROR – Error in setting slot into sleep/active modes

### Adpd400xDrvDataReadyCallback

**Prototype**

void Adpd400xDrvDataReadyCallback(void (\*pfADPDDataReady)());

**Description**

Registers the data ready callback.

**Parameters**

**Name:** pfADPDDataReady

**Type:** void \*

**Direction:** Input/Output

**Description:** Function pointer callback for the register data

**Return value**

None

### Adpd400xISR

**Prototype**

void Adpd400xISR();

**Description**

Interrupt service routine

**Parameters**

None

**Return value**

None

### Adpd400xDrvSetParameter

**Prototype**

int16\_t Adpd400xDrvSetParameter(Adpd400xCommandStruct\_t eCommand, uint8\_t nPar, uint16\_t nValue);

**Description**

Sets the configuration parameters for the device. The watermark for the FIFO is the option available now for configuring the device.

The test data command is used for internal testing only.

**Parameters**

**Name:** eCommand

**Type:** Adpd400xCommandStruct\_t

**Direction:** Input/Output

**Description:** The command for the desired parameter to set

**Name:** nPar

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** This parameter is not used now

**Name:** nValue

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** The value to be set for the parameter

**Return value**

ADPD400xDrv\_SUCCESS – Setting the parameter successful

ADPD400xDrv\_ERROR – Error in setting the parameter

### Adpd400xDrvGetParameter

**Prototype**

int16\_t Adpd400xDrvSetParameter(Adpd400xCommandStruct\_t eCommand, uint8\_t nPar, uint16\_t \*pnValue);

**Description**

Gets the configuration parameters for the device. The parameters that are obtained includes:

* Watermark
* Output Data Rate
* Fifo Level
* Time Gap
* Latest Slot data size
* Current slot data size
* Total slots data size
* If the current slot is active
* Check if this slot is selected
* Highest slot selected
* Number of active channels for this slot

**Parameters**

**Name:** eCommand

**Type:** Adpd400xCommandStruct\_t

**Direction:** Input/Output

**Description:** The command for the desired parameter to get

**Name:** nPar

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** This parameter specifies the slot number

**Name:** pnValue

**Type:** uint16\_t \*

**Direction:** Input/Output

**Description:** The value to be set for the parameter

**Return value**

ADPD400xDrv\_SUCCESS – Getting the parameter successful

ADPD400xDrv\_ERROR – Error in getting the parameter

### Adpd400xDrvReadFifoData

**Prototype**

int16\_t Adpd400xDrvReadFifoData(uint8\_t \*pnData, uint16\_t nDataSetSize);

**Description**

This function does the read of data from the ADPD4100 FIFO

**Parameters**

**Name:** pnData

**Type:** uint8\_t \*

**Direction:** Input/Output

**Description:** Pointer location to read data into

**Name:** nDataSetSize

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Dataset size to read

**Return value**

ADPD400xDrv\_SUCCESS – Read from FIFO successful

ADPD400xDrv\_ERROR – Error in reading from FIFO

### Adpd400xDrvReadRegData

**Prototype**

int16\_t Adpd400xDrvReadRegData(uint32\_t \*pnData, ADPD400xDrv\_SlotNum\_t nSlotNum, uint8\_t nSignalDark, uint8\_t nChNum);

**Description**

This function does the read of data from the ADPD4100 register for a particular slot

**Parameters**

**Name:** pnData

**Type:** uint32\_t \*

**Direction:** Input/Output

**Description:** Pointer location to read data into

**Name:** nSlotNum

**Type:** ADPD400xDrv\_SlotNum\_t

**Direction:** Input/Output

**Description:** Slot number to read data from

**Name:** nSignalDark

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Dark/Signal flag

**Name:** nChNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Channel number from the slot to read

**Return value**

ADPD400xDrv\_SUCCESS – Read from register of a slot successful

ADPD400xDrv\_ERROR – Error in reading from register for a slot

### Adpd400xDrvSetLedCurrent

**Prototype**

int16\_t Adpd400xDrvSetLedCurrent(uint16\_t nLedCurrent, ADPD400xDrv\_LedId\_t nLedId, ADPD400xDrv\_SlotNum\_t nSlotNum);

**Description**

This function sets the LED current for the LED current to the slot

**Parameters**

**Name:** nLedCurrent

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** LED current to set to. 0: disable, max is 0x7F = 200mA (for details of this setting, see DataSheet)

**Name:** nLedId

**Type:** ADPD400xDrv\_LedId\_t

**Direction:** Input/Output

**Description:** LED number. Check the LED connected to each slot on the platform

**Name:** nSlotNum

**Type:** ADPD400xDrv\_SlotNum\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Return value**

ADPD400xDrv\_SUCCESS – Setting LED current successful

ADPD400xDrv\_ERROR – Error in setting LED current

### Adpd400xDrvGetLedCurrent

**Prototype**

int16\_t Adpd400xDrGetLedCurrent(uint16\_t \*pLedCurrent, ADPD400xDrv\_LedId\_t nLedId, ADPD400xDrv\_SlotNum\_t nSlotNum);

**Description**

This function sets the LED current for the LED current to the slot

**Parameters**

**Name:** pLedCurrent

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Pointer location to get LED current

**Name:** nLedId

**Type:** ADPD400xDrv\_LedId\_t

**Direction:** Input/Output

**Description:** LED number. Check the LED connected to each slot on the platform

**Name:** nSlotNum

**Type:** ADPD400xDrv\_SlotNum\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Return value**

ADPD400xDrv\_SUCCESS – Getting LED current successful

ADPD400xDrv\_ERROR – Error in getting LED current

### Adpd400xDrvSoftReset

**Prototype**

int16\_t Adpd400xDrvSoftReset(void);

**Description**

This function does a soft reset of the ADPD4100 device

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Soft reset of ADPD4100 successful

ADPD400xDrv\_ERROR – Error in soft resetting of ADPD4100

### \_Adpd400xDrvInit

**Prototype**

static void \_Adpd400xDrvInit(void);

**Description**

This function initializes the driver.

**Parameters**

None

**Return value**

None

### \_Adpd400xDrvSetInterrupt

**Prototype**

static int16\_t \_Adpd400xDrvSetInterrupt();

**Description**

This function sets the FIFO interrupt mode.

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Setting of FIFO interrupt mode successful

ADPD400xDrv\_ERROR – Error in setting FIFO interrupt mode

### \_Adpd400xDrvSetIdleMode

**Prototype**

static int16\_t \_Adpd400xDrvSetIdleMode();

**Description**

This function sets the device to Idle mode.

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Setting to Idle mode successful

ADPD400xDrv\_ERROR – Error in setting to Idle mode

### \_Adpd400xDrvGetSlotInfo

**Prototype**

static void \_Adpd400xDrvGetSlotInfo();

**Description**

This function gets the status of the twelve slots.

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Getting the status of slots successful

ADPD400xDrv\_ERROR – Error in getting the status of slots

### \_Adpd400xDrvGetDataOuputRate

**Prototype**

static void \_Adpd400xDrvGetDataOuputRate();

**Description**

This function gets the output data rate.

**Parameters**

None

**Return value**

ADPD400xDrv\_SUCCESS – Getting the output data rate successful

ADPD400xDrv\_ERROR – Error in getting the output data rate

### \_Adpd400xDrvSlotSaveCurrentSetting

**Prototype**

static void \_Adpd400xDrvSlotSaveCurrentSetting(uint8\_t nSlotNum);

**Description**

This function saves the current setting

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Return value**

ADPD400xDrv\_SUCCESS – Saving the current setting successful

ADPD400xDrv\_ERROR – Error in saving current setting

### \_Adpd400xDrvSlotApplyPreviousSetting

**Prototype**

static void \_Adpd400xDrvSlotApplyPreviousSetting(uint8\_t nSlotNum);

**Description**

This function is used to apply the previous LED settings

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Return value**

ADPD400xDrv\_SUCCESS – Applying the previous LED settings successful

ADPD400xDrv\_ERROR – Error in applying the previous LED settings

### \_Adpd400xDrvSlotApplySkipSetting

**Prototype**

static void \_Adpd400xDrvSlotApplySkipSetting(uint8\_t nSlotNum);

**Description**

This function restores the previous LED settings

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Return value**

ADPD400xDrv\_SUCCESS – Restoring the previous LED settings successful

ADPD400xDrv\_ERROR – Error in restoring the previous LED settings

### \_Adpd400xDrvSetSlotSize

**Prototype**

static void \_Adpd400xDrvSetSlotSize(uint8\_t nSlotNum, uint16\_t nSlotFormat);

**Description**

This function sets the slot data size

**Parameters**

**Name:** nSlotNum

**Type:** uint8\_t

**Direction:** Input/Output

**Description:** Slot Number of the LED

**Name:** nSlotFormat

**Type:** uint16\_t

**Direction:** Input/Output

**Description:** Slot Format of the data

**Return value**

ADPD400xDrv\_SUCCESS – Setting the slot data size successful

ADPD400xDrv\_ERROR – Error in setting the slot data size

### \_FifoLevel

**Prototype**

static uint16\_t \_FifoLevel(void;

**Description**

This function finds the FIFO level

**Parameters**

None

**Return value**

Fifo level

## API Data Types

### adpd400xDrv\_slot\_t

typedef struct \_adpd400xDrv\_slot\_t {

uint8\_t activeSlot; //!< Active slot

uint8\_t pre\_activeSlot; //!< Previous Active slot

uint16\_t slotFormat; //!< Dark,Sig,Lit,Ttl bytes of each slot

uint8\_t channelNum; //!< Active channel for each slot

uint8\_t decimation;

} adpd400xDrv\_slot\_t;

**Description**

The adpd400xDrv\_slot\_t structure contains the various parameters related to a slot of ADPD4100.

**Fields**

* adpd400xDrv\_slot\_t.activeSlot specifies if the slot is active
* adpd400xDrv\_slot\_t.pre\_activeslot stores the previous active slot
* adpd400xDrv\_slot\_t.slotFormat specifies the format of the slot data (dark/sig/lit)
* adpd400xDrv\_slot\_t.channelNum indicates the active the channels for the slot
* adpd400xDrv\_slot\_t.decimation indicates the decimation factor for the slot

## Enums

### ADPD400xDrv\_Operation\_Mode\_t

typedef enum {

ADPD400xDrv\_MODE\_IDLE = 0,

ADPD400xDrv\_MODE\_PAUSE,

ADPD400xDrv\_MODE\_PWR\_OFF,

ADPD400xDrv\_MODE\_SAMPLE

} ADPD400xDrv\_Operation\_Mode\_t;

**Description**

Indicates the operation modes of the ADPD4100 device

### ADPD400XDrv\_FIFO\_SIZE\_t

typedef enum {

ADPD400xDrv\_SIZE\_0 = 0x00,

ADPD400xDrv\_SIZE\_8 = 0x01,

ADPD400xDrv\_SIZE\_16 = 0x02,

ADPD400xDrv\_SIZE\_24 = 0x03,

ADPD400xDrv\_SIZE\_32 = 0x04,

} ADPD400XDrv\_FIFO\_SIZE\_t;

**Description**

Enumerates the various data sizes of the FIFO data. This can be a byte as minimum and 4 bytes as maximum.

### Adpd400xCommandStruct\_t

typedef enum {

ADPD400x\_WATERMARKING = 0,

ADPD400x\_FIFOLEVEL,

ADPD400x\_OUTPUTDATARATE,

ADPD400x\_TIMEGAP,

ADPD400x\_LATEST\_SLOT\_DATASIZE,

ADPD400x\_THIS\_SLOT\_DATASIZE,

ADPD400x\_SUM\_SLOT\_DATASIZE,

ADPD400x\_IS\_SLOT\_ACTIVE,

ADPD400x\_IS\_SLOT\_SELECTED,

ADPD400x\_HIGHEST\_SLOT\_NUM,

ADPD400x\_THIS\_SLOT\_CHANNEL\_NUM,

ADPD400x\_TEST\_DATA

} Adpd400xCommandStruct\_t;

**Description**

This shows the various commands used for setting or getting parameters of the ADPD4100 device.

### ADPDDrvCl\_SignalDark\_t

typedef enum {

ADPD400xDrv\_SIGNAL = 0x00,

ADPD400xDrv\_DARK

} ADPDDrvCl\_SignalDark\_t;

**Description**

This structure lists the signal and dark parts of the data

### ADPD400xDrv\_SlotNum\_t

typedef enum {

ADPD400xDrv\_SLOTA = 0x00,

ADPD400xDrv\_SLOTB,

ADPD400xDrv\_SLOTC,

ADPD400xDrv\_SLOTD,

ADPD400xDrv\_SLOTE,

ADPD400xDrv\_SLOTF,

ADPD400xDrv\_SLOTG,

ADPD400xDrv\_SLOTH,

ADPD400xDrv\_SLOTI,

ADPD400xDrv\_SLOTJ,

ADPD400xDrv\_SLOTK,

ADPD400xDrv\_SLOTL

} ADPD400xDrv\_SlotNum\_t;

**Description**

This structure maps the slots on the ADPD4100 device

### ADPD400xDrv\_LedId\_t

typedef enum {

ADPD400xDrv\_LED\_OFF = 0x00,

ADPD400xDrv\_LED1,

ADPD400xDrv\_LED2,

ADPD400xDrv\_LED3,

ADPD400xDrv\_LED4

} ADPD400xDrv\_LedId\_t;

**Description**

This structure maps the LEDs available on the ADPD4100 device

### Adpd400xComMode\_t

typedef enum {

ADPD400x\_I2C\_BUS, /\*\*< enum value 0 \*/

ADPD400x\_SPI\_BUS, /\*\*< enum value 1 \*/

ADPD400x\_UNKNOWN\_BUS /\*\*< enum value 2 \*/

} Adpd400xComMode\_t;

**Description**

This enumeration shows the different data interface modes available on ADPD410X devices