ADI VSM STUDY WATCH DRIVER REFERENCE GUIDE

ANALOG DEVICES, INC.

www.analog.com

REV 1.0.0,FEB 2021

Table of Contents

List of Figures

List of Tables

Copyright, Disclaimer & Trademark Statements

Copyright Information

Disclaimer

Trademark and Service Mark Notice

1 Introduction

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

1.1 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.

1.2 Organization of this Guide

Section 1: this section contains the introduction

Section 2: lists specifications of the devices

Section 3: ADPD4100 driver API guide

Section 4: ADXL362 driver API guide

1.3 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

1.4 References

I. ADPD4100 Data sheet

II. ADXL362 Data sheet

1.5 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.

2 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.

2.1 Version Information

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

3 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.

3.1 API Functions

3.1.1 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

3.1.2 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

3.1.3 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

3.1.4 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

3.1.5 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

3.1.6 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

3.1.7 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

3.1.8 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

3.1.9 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

3.1.10 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

3.1.11 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

3.1.12 Adpd400xISR

Prototype
```
void Adpd400xISR();
```

Description
Interrupt service routine

Parameters
None

Return value
None

3.1.13 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

3.1.14 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

3.1.15 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

3.1.16 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

3.1.17 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

3.1.18 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

3.1.19 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

3.1.20 _Adpd400xDrvInit

Prototype

```
static void _Adpd400xDrvInit(void);
```

Description

This function initializes the driver.

Parameters

None

Return value

None

3.1.21 _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

3.1.22 _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

3.1.23 _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

3.1.24 _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

3.1.25 _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

3.1.26 _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

3.1.27 _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

3.1.28 _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

3.1.29 _FifoLevel

Prototype

```
static uint16_t _FifoLevel(void;
```

Description
This function finds the FIFO level

Parameters
None

Return value
Fifo level

3.2 API Data Types

3.2.1 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

3.3 Enums

3.3.1 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

3.3.2 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.

3.3.3 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.

3.3.4 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

3.3.5 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

3.3.6 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

3.3.7 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