Digilent chipKIT ACL Reference Manual

Revision: June 27, 2012

1300 NE Henley Court, Suite 3
Pullman, WA 99163
(509) 334 6306 Voice | (509) 334 6300 Fax

Introduction

The Digilent PmodACL is a 3-axis digital accelerometer module driven by the Analog Devices ADXL345 accelerometer.
Digilent provides an Arduino driver library for this device, covering its most important functionality.
This document provides an overview of this library.
Read Overview chapter for an overview of PmodACL
Read Library implementation for details about ACL library implementation.
In the end, the document describes the Library usage.

Overview

The accelerometer is a device that can be accessed via an SPI or I2C interface. Read and write operations can be performed in a multi-byte mode. The device allows configuration, acceleration reading, as well as interrupts related capabilities.
This document refers to data specific to Analog Devices ADXL345 accelerometer (register names, register bits). For more information refer to the PmodACL Reference Manual available for download from the Digilent web site (www.digilentinc.com) and to the Analog Devices ADXL345 datasheet.

Library implementation

Library functions

void begin(uint8_t bAccessType)
Parameters

- uint8_t bAccessType – The parameter indicating the access type, which can be SPI or I2C. It can be one of the parameters from the following list:

Table 1. List of Parameters for begin function

Value	Name	Port on CerebotMX4cK	Port on CerebotMX3cK	Port on CerebotMX7cK
0	PAR_ACCESS_DSPI0	SPI1(J1)	SPI2(JE)	SPI1(JD)
1	PAR_ACCESS_DSPI1	SPI2(JB)	SPI1(JD)	SPI3(JE)
2	PAR_ACCESS_DSPI2	-	-	SPI4(JF)
10	PAR_ACCESS_I2C	I2C#1 (J2)	I2C#1 (J2)	I2C#2(J8)

The port corresponding to each DSPI (0, 1, 2) or I2C is specified in each board's reference manual.

This function performs the required ACL initialization tasks:

- Initializes and configures the SPI or I2C communication instance. If SPI is selected, it sets the default SPI frequency to 1 MHz.
- Configures the required signals as outputs
- Other initialization tasks

void end()
This function performs the required ACL deinitialization tasks.

void ReadAccelG(float &dAclXg, float &dAclYg, float &dAclZg)
Parameters

- float &dAclXg - the output parameter that will receive acceleration on X axis (in "g")
- float &dAclYg - the output parameter that will receive acceleration on Y axis (in "g")
- float &dAclZg - the output parameter that will receive acceleration on Z axis (in "g")

This function is the main function used for acceleration reading, providing the 3 current acceleration values in “g”.

- It reads simultaneously the acceleration on three axes in a buffer of 6 bytes using the ReadRegister function
- For each of the three axes, combines the two bytes in order to get a 10-bit value
- For each of the three axes, converts the 10-bit value to the value expressed in “g”, considering the currently selected g range

void ReadAccel(int16_t &iAclXg, int16_t &iAclYg, int16_t &iAclZg)
Parameters

- int16_t &iAclX - the output parameter that will receive acceleration on X axis - 10 bits signed value
- int16_t &iAclY - the output parameter that will receive acceleration on Y axis - 10 bits signed value
- int16_t &iAclZ - the output parameter that will receive acceleration on Z axis - 10 bits signed value

This function provides the 3 "raw" 10-bit values read from the accelerometer.

- It reads simultaneously the acceleration on three axes in a buffer of 6 bytes using the ReadRegister function
- For each of the three axes, combines the two bytes in order to get a 10-bit value

uint8_t SetMeasure(bool fMeasure)

Parameters

- bool fMeasure – the value to be set for MEASURE bit of POWER_CTL register

This function sets the MEASURE bit of POWER_CTL register. This toggles between measurement and standby mode.

float GetMeasure(uint8_t bAxisParam, uint8_t *pErr)

Return value:
bool – the value of the MEASURE bit of POWER_CTL register.

This function returns the value of MEASURE bit of POWER_CTL register.

uint8_t SetGRange(uint8_t bGRangePar)

Parameters

- uint8_t bDataFormatGRangePar - the parameter specifying the g range. Can be one of the parameters from the following list:

Table 2. List of parameters for SetGRange function

Value	Name	Corresponding to:
0	PAR_GRANGE_PM2G	+/- 2g
1	PAR_GRANGE_PM4G	+/- 4g
2	PAR_GRANGE_PM8G	+/- 8g
3	PAR_GRANGE_PM16G	+/- 16g

The function sets the appropriate g range bits in the DATA_FORMAT register. The accepted argument values are between 0 and 3.
If the argument is within the accepted values range, it sets the g range bits in DATA_FORMAT register and ACL_ERR_SUCCESS status is returned.
If value is outside this range no value is set.

uint8_t GetGRange()

Return value

- uint8_t - the value specifying the g Range parameter. Can be one of the values from the following list:

Table 3. List of parameters for ACLGetDataFormatGRangePar function

Value	Name	Corresponding to:
0	PAR_GRANGE_PM2G	+/- 2g
1	PAR_GRANGE_PM4G	+/- 4g
2	PAR_GRANGE_PM8G	+/- 8g
3	PAR_GRANGE_PM16G	+/- 16g

The function returns the value specifying the g range parameter. It relies on the data in DATA_FORMAT register.

uint8_t SetOffsetG(uint8_t bAxisParam, float dOffset)

Parameters

- uint8_t bAxisParam byte indicating the axis whose offset will be set. Can be one of:
  - PAR_AXIS_X - indicating X axis
  - PAR_AXIS_Y - indicating Y axis
  - PAR_AXIS_Z - indicating Z axis

- float dOffsetX – the offset for X axis in “g”.

This function sets the specified axis offset, the value being given in “g”. The accepted argument values are between -2g and +2g.
If argument is within the accepted values range, its value is quantified in the 8-bit offset register using a scale factor of 15.6 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range or if bAxisParam parameter is outside 0 - 3 range, the function does nothing.

float GetOffsetG(uint8_t bAxisParam, uint8_t *pErr)

Parameters

- uint8_t bAxisParam - byte indicating the axis whose offset will be returned. Can be one of:
  - PAR_AXIS_X - indicating X axis
  - PAR_AXIS_Y - indicating Y axis
  - PAR_AXIS_Z - indicating Z axis

Return value:

float – the offset in “g” for the specified axis.

This function returns the offset, in “g”, for the specified axis.
It converts the 8-bit value quantified in the offset register into a value expressed in “g”, using a scale factor of 15.6 mg/LSB.

void CalibrateOneAxisGravitational(uint8_t bAxisInfo)

Parameters

uint8_t bAxisInfo - Parameter specifying axes orientation. Can be one of the following:

Table 4. List of parameters for bAxisInfo parameter of ACLCalibrateOneAxisGravitational function

Value	Name	Meaning
0	PAR_AXIS_XP	x axis is oriented in the gravitational direction.
1	PAR_AXIS_XN	x axis is oriented in the opposite gravitational direction.
2	PAR_AXIS_YP	y axis is oriented in the gravitational direction.
3	PAR_AXIS_YN	y axis is oriented in the opposite gravitational direction.
4	PAR_AXIS_ZP	z axis is oriented in the gravitational direction.
5	PAR_AXIS_ZN	z axis is oriented in the opposite gravitational direction.

This function performs the calibration of the accelerometer by setting the offset registers in the following manner: computes the correction factor that must be loaded in the offset registers so that the acceleration readings are:

1 for the gravitational axis, if positive orientation
-1 for the gravitational axis, if negative orientation
0 for the other axes

The accepted argument values are between 0 and 5.
If the argument value is outside this range, the function does nothing.

Library usage

This section of the document describes the way the library is used:

- The PmodACL should be plugged in one of the SPI or I2C connectors.

Table 5. List of possible connections for PmodACL (for CerebotMX4cK board)

Connection	Connector
SPI1	J1
SPI2	JB
I2C	I2C#1

- When I2C is used, make sure that PmodACL SS line is either disconnected either connected to Vcc.
- Copy the library files according to the README.txt file.
- In the sketch, include the ACL library header file
```
#include <ACL.h>
```
- In the sketch, include the DSPI and Wire libraries header files. They are needed to connect the accelerometer via SPI or I2C to the board.
```
#include <Wire.h>
#include <DSPI.h>
```
- In the sketch, instantiate one library object called, for example, myACL
```
ACL myACL;
```
- In the sketch, use library functions by calls such as:
```
myACL.SetMeasure(true);
```

Simple Demo

In order to run this demo:

- place the library and sketch as explained in the Library usage section.
- Connect the board using SPI1 (J1 connector).
- Upload the application and start Serial Monitor.

This demo performs the following operations:

- In the setup() function:
  - Initializes the ACL library and Serial interface (in order to display information on the Serial terminal).
  - Performs the PmodACL calibration
- In the loop() function:
  - Reads the accelerometer values
  - Displays the read values using the Serial interface