Digilent chipKIT ACL Reference Manual

Revision: November 29, 2011

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. This document provides an overview of this library operation.
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 functionality, as well as interrupts 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

Errors

This list shows the possible errors returned by ACL functions:

Table 1. List of errors

Value	Name	Description
0	ACL_ERR_SUCCESS	The action completed successfully
1	ACL_ERR_ARG_RANGE_PM2G	The argument is not within -2g, 2g range
2	ACL_ERR_ARG_RANGE_016G	The argument is not within 0, 16g range
3	ACL_ERR_ARG_RANGE_0160MS	The argument is not within 0, 160ms range
4	ACL_ERR_ARG_RANGE_0320MS	The argument is not within 0, 320ms range
5	ACL_ERR_ARG_RANGE_0128S	The argument is not within 0, 1.28s range
6	ACL_ERR_ARG_RANGE_015	The argument is not within 0, 15 range
7	ACL_ERR_ARG_RANGE_DF_03	The argument is not within 0, 3 range
8	ACL_ERR_ARG_RANGE_FIFO_03	The argument is not within 0, 3 range
9	ACL_ERR_ARG_RANGE_WU_03	The argument is not within 0, 3 range
10	ACL_ERR_ARG_RANGE_05	The argument is not within 0, 5 range
32	ACL_ERR_ARG_AINTNO01_BIT	The argument is not within 0, 1 range (ACL Interrupt no)
64	ACL_ERR_ARG_EINTNO04_BIT	The argument is not within 0, 4 range

		(External Interrupt no)
128	ACL_ERR_ARG_ACT_BIT	The argument is not within 0, 1 range (Active)

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 2. List of Parameters for begin function

Value	Name
0	PAR_ACCESS_SPI0
1	PAR_ACCESS_SPI1
2	PAR_ACCESS_I2C

This function performs the required ACL initialization tasks:

- Initializes and configures the SPI or I2C communication instance. 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 *pdAclXg, float *pdAclYg, float *pdAclZg)
Parameters

- float *pdAclXg – the pointer to the variable that will receive acceleration on X axis (in "g")
- float *pdAclYg – the pointer to the variable that will receive acceleration on Y axis (in "g")
  float *pdAclZg – the pointer to the variable 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 ACLReadRegister
- For each of the three values, combines the two bytes in order to get a 10-bit value
- For each of the three values, converts the 10-bit value to the value expressed in “g”, considering the currently selected g range

float ReadAccelXG()
Return value:

- the value of the acceleration on X axis (in "g")

This function is an alternate function used for acceleration reading (the main function for this is ACLReadAccelG), providing the current acceleration values in “g” on X axis.

- It reads the acceleration on X axis in a buffer of 2 bytes (2 bytes variable) using the ACLReadRegister
- Combines the two bytes in order to get a 10-bit value
- Converts the 10-bit value to the value in “g”, considering the currently selected g range.

When using this function, you must know that it only reads the X axis acceleration. Thus, advancing position in FIFO buffers is done unequally for the 3 axes. The recommended way is to use the ACLReadAccelG function.

float ReadAccelYG()
Return value:
- the value of the acceleration on Y axis (in "g")

This function is an alternate function used for acceleration reading (the main function for this is ACLReadAccelG), providing the current acceleration values in “g” on Y axis.
- It reads the acceleration on Y axis in a buffer of 2 bytes (2 bytes variable) using the ACLReadRegister
- Combines the two bytes in order to get a 10-bit value
- Converts the 10-bit value to the value in “g”, considering the currently selected g range using ACLConvertReadingToValueG

When using this function, you must know that it only reads the Y axis acceleration. Thus, advancing position in FIFO buffers is done unequally for the 3 axes. The recommended way is to use the ACLReadAccelG function.

float ReadAccelZG()
Return value:
- the value of the acceleration on Z axis (in "g")

This function is an alternate function used for acceleration reading (the main function for this is ACLReadAccelG), providing the current acceleration values in “g” on Z axis.
- It reads the acceleration on Z axis in a buffer of 2 bytes (2 bytes variable) using the ACLReadRegister
- Combines the two bytes in order to get a 10-bit value
- Converts the 10-bit value to the value in “g”, considering the current selected g range using ACLConvertReadingToValueG

When using this function, you must know that it only reads the Z axis acceleration. Thus, advancing position in FIFO buffers is done unequally for the 3 axes. The recommended way is to use ACLReadAccelG function.

Other functions

uint8_t GetDevID()

This function returns the Device ID for ADXL345 accelerometer, which is 0xE5 (see datasheet).

uint8_t SetOffsetXG(float dOffsetX)

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

Return value:
- *ACL_ERR_SUCCESS - The action completed successfully*
- *ACL_ERR_ARG_RANGE_PM2G - The argument is not within +/- 2g range*

See Errors for the errors list.

This function sets the X 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 OFSZ register using a scale factor of 15.6 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_PM2G is returned and no value is set.

float PmodACL::ACLGetOffsetXG()

Return value:
float – the offset for X axis in “g”.

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

uint8_t SetOffsetYG(float dOffsetY)

Parameters
- float dOffsetY – the offset for Y axis in “g”.

Return value:
- *ACL_ERR_SUCCESS - The action completed successfully*
- *ACL_ERR_ARG_RANGE_PM2G - The argument is not within +/- 2g range*

See Errors for the errors list.

This function sets the Y 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 OFSZ register using a scale factor of 15.6 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_PM2G is returned and no value is set.

float PmodACL::ACLGetOffsetYG()
Return value:
float – the offset for Y axis in “g”.

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

uint8_t SetOffsetZG(float dOffsetZ)

Parameters
- float dOffsetZ – the offset for Z axis in “g”.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_PM2G - The argument is not within +/- 2g range
See Errors for the errors list.

This function sets the Z 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 OFSZ register using a scale factor of 15.6 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_PM2G is returned and no value is set.

float PmodACL::ACLGetOffsetZG()
Return value:
float – the offset for Z axis in “g”.

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

uint8_t SetBWRateOutputRatePar(uint8_t bOutputRatePar)
Parameters
- uint8_t bOutputRatePar – a parameter selecting one of the options from the following list.

Table 3 List of values for bOutputRatePar parameter in SetBWRateOutputRatePar function

Value	Name	Note
0	PAR_OUTPUTRATE0_10Hz	Parameter Output Data Rate : 0.10 Hz, used to access Rate in BW_RATE register
1	PAR_OUTPUTRATE0_20Hz	Parameter Output Data Rate : 0.20 Hz, used to access Rate in BW_RATE register
2	PAR_OUTPUTRATE0_39Hz	Parameter Output Data Rate : 0.39 Hz, used to access Rate in BW_RATE register
3	PAR_OUTPUTRATE0_78Hz	Parameter Output Data Rate : 0.78 Hz, used to access Rate in BW_RATE register
4	PAR_OUTPUTRATE1_56Hz	Parameter Output Data Rate : 1_56 Hz, used to access Rate in BW_RATE register
5	PAR_OUTPUTRATE3_13Hz	Parameter Output Data Rate : 3.13 Hz, used to access Rate in BW_RATE register
6	PAR_OUTPUTRATE6_25Hz	Parameter Output Data Rate : 6.25 Hz, used to access Rate in BW_RATE register
7	PAR_OUTPUTRATE12_50Hz	Parameter Output Data Rate : 12.5 Hz, used to access Rate in BW_RATE register
8	PAR_OUTPUTRATE25_00Hz	Parameter Output Data Rate : 25 Hz, used to access Rate in BW_RATE register

9	PAR_OUTPUTRATE50_00Hz	Parameter Output Data Rate : 50 Hz, used to access Rate in BW_RATE register
10	PAR_OUTPUTRATE100_00Hz	Parameter Output Data Rate : 100 Hz, used to access Rate in BW_RATE register
11	PAR_OUTPUTRATE200_00Hz	Parameter Output Data Rate : 200 Hz, used to access Rate in BW_RATE register
12	PAR_OUTPUTRATE400_00Hz	Parameter Output Data Rate : 400 Hz, used to access Rate in BW_RATE register
13	PAR_OUTPUTRATE800_00Hz	Parameter Output Data Rate : 800 Hz, used to access Rate in BW_RATE register
14	PAR_OUTPUTRATE1600_00Hz	Parameter Output Data Rate : 1600 Hz, used to access Rate in BW_RATE register
15	PAR_OUTPUTRATE3200_00Hz	Parameter Output Data Rate : 3200 Hz, used to access Rate in BW_RATE register

Return value:

- ACL_ERR_SUCCESS - The action completed successfully

- ACL_ERR_ARG_RANGE_015 - The argument is not within 0 - 15 range

See Errors for the errors list.

The accepted argument values are between 0 and 15. If the argument is within the accepted values range, the function sets the Rate bits in the BW_RATE register according to the parameter and ACL_ERR_SUCCESS is returned.
If the value is outside this range, ACL_ERR_ARG_RANGE_015 is returned and no value is set.

uint8_t GetBWRateOutputRatePar()

Return value

- uint8_t – the value specifying the output rate parameter. Can be one of the values from the following list:

Table 4. List of values returned by GetBWRateOutputRatePar function

Value	Name	Note
0	PAR_OUTPUTRATE0_10Hz	Parameter Output Data Rate : 0.10 Hz, used to access Rate in BW_RATE register
1	PAR_OUTPUTRATE0_20Hz	Parameter Output Data Rate : 0.20 Hz, used to access Rate in BW_RATE register
2	PAR_OUTPUTRATE0_39Hz	Parameter Output Data Rate : 0.39 Hz, used to access Rate in BW_RATE register
3	PAR_OUTPUTRATE0_78Hz	Parameter Output Data Rate : 0.78 Hz, used to access Rate in BW_RATE

		register
4	PAR_OUTPUTRATE1_56Hz	Parameter Output Data Rate : 1_56 Hz, used to access Rate in BW_RATE register
5	PAR_OUTPUTRATE3_13Hz	Parameter Output Data Rate : 3.13 Hz, used to access Rate in BW_RATE register
6	PAR_OUTPUTRATE6_25Hz	Parameter Output Data Rate : 6.25 Hz, used to access Rate in BW_RATE register
7	PAR_OUTPUTRATE12_50Hz	Parameter Output Data Rate : 12.5 Hz, used to access Rate in BW_RATE register
8	PAR_OUTPUTRATE25_00Hz	Parameter Output Data Rate : 25 Hz, used to access Rate in BW_RATE register
9	PAR_OUTPUTRATE50_00Hz	Parameter Output Data Rate : 50 Hz, used to access Rate in BW_RATE register
10	PAR_OUTPUTRATE100_00Hz	Parameter Output Data Rate : 100 Hz, used to access Rate in BW_RATE register
11	PAR_OUTPUTRATE200_00Hz	Parameter Output Data Rate : 200 Hz, used to access Rate in BW_RATE register
12	PAR_OUTPUTRATE400_00Hz	Parameter Output Data Rate : 400 Hz, used to access Rate in BW_RATE register
13	PAR_OUTPUTRATE800_00Hz	Parameter Output Data Rate : 800 Hz, used to access Rate in BW_RATE register
14	PAR_OUTPUTRATE1600_00Hz	Parameter Output Data Rate : 1600 Hz, used to access Rate in BW_RATE register
15	PAR_OUTPUTRATE3200_00Hz	Parameter Output Data Rate : 3200 Hz, used to access Rate in BW_RATE register

This function returns a value corresponding to the Output Rate bits in the BW_RATE register.

void Set BWRateLowPowerBit(bool fLowPower)

Parameters

- bool fLowPower – the boolean value to be set to the low power bit.

This function sets the LOW_POWER bit of the BW_RATE register.

boolGetBWRateLowPowerBit()

Return value

- bool – the boolean value of the LOW_POWER bit of the BW_RATE register.

This function returns the LOW_POWER bit of the BW_RATE register.

void SetPowerControlLinkBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the Link bit of the *POWER_CTL* register.

This function sets the Link bit of the *POWER_CTL* register.

boolGetPowerControlLinkBit()

Return value
- bool – the boolean value of the Link bit of the *POWER_CTL* register.

This function returns the Link bit of the *POWER_CTL* register.

void SetPowerControlAutoSleepBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to AUTO_SLEEP bit of the *POWER_CTL* register.

This function sets the AUTO_SLEEP bit of the *POWER_CTL* register.

boolGetPowerControlAutoSleepBit()

Return value
- bool – the boolean value of the AUTO_SLEEP bit of the *POWER_CTL* register.

This function returns the AUTO_SLEEP bit value of the *POWER_CTL* register.

void SetPowerControlMeasureBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the Measure bit of the *POWER_CTL* register.

This function sets the Measure bit of the *POWER_CTL* register.

boolGetPowerControlMeasureBit()

Return value
- bool – the boolean value of the Measure bit of the *POWER_CTL* register.

This function returns the Measure bit of the *POWER_CTL* register.

void SetPowerControlSleepBit(bool fValue)
Parameters
- bool fValue – the boolean value to be set to the Sleep bit in the *POWER_CTL* register.

This function sets the Sleep bit of the *POWER_CTL* register.

boolGetPowerControlSleepBit()

Return value

- bool – the boolean value of the Sleep bit of the *POWER_CTL* register.

This function returns the Sleep bit in the *POWER_CTL* register.

uint8_t SetPowerControlWakeupFreqPar(uint8_t bPowerControlWakeupFreqPar)

Parameters

- uint8_t bPowerControlWakeupFreqPar - the parameter specifying the wakeup frequency. Can be one of the parameters in the following list:

Table 5. List of values for bOutputRatePar parameter in SetBWRateOutputRatePar function ACLSetPowerControlWakeupFreqPar function

Value	Name	Note
0	PAR_WAKEUP_FREQ8Hz	Parameter Wakeup rate : 8 Hz, used to access Wakeup bits in *POWER_CTL* register
1	PAR_WAKEUP_FREQ4Hz	Parameter Wakeup rate : 4 Hz, used to access Wakeup bits in *POWER_CTL* register
2	PAR_WAKEUP_FREQ2Hz	Parameter Wakeup rate : 2 Hz, used to access Wakeup bits in *POWER_CTL* register
3	PAR_WAKEUP_FREQ1Hz	Parameter Wakeup rate : 1 Hz, used to access Wakeup bits in *POWER_CTL* register

Return value:

- ACL_ERR_SUCCESS - The action completed successfully

- ACL_ERR_ARG_RANGE_DF_03 - The argument is not within 0 - 3 range

See Errors for the errors list.

This function sets the appropriate wakeup frequency bits in POWER_CTL register. The accepted argument values are between 0 and 3.
If argument is within the accepted values range, it sets the appropriate wakeup frequency bits in the POWER_CTL register and ACL_ERR_SUCCESS status is returned. If the value is outside this range, ACL_ERR_ARG_RANGE_DF_03 is returned and no value is set.

uint8_t GetPowerControlWakeupFreqPar()

Return value

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

Table 6. List of parameters for ACLGetPowerControlWakeupFreqPar function

Value	Name	Note
0	PAR_WAKEUP_FREQ8Hz	Parameter Wakeup rate : 8 Hz, used to access Wakeup bits in *POWER_CTL* register
1	PAR_WAKEUP_FREQ4Hz	Parameter Wakeup rate : 4 Hz, used to

		access Wakeup bits in *POWER_CTL* register
2	PAR_WAKEUP_FREQ2Hz	Parameter Wakeup rate : 2 Hz, used to access Wakeup bits in *POWER_CTL* register
3	PAR_WAKEUP_FREQ1Hz	Parameter Wakeup rate : 1 Hz, used to access Wakeup bits in *POWER_CTL* register

The function returns the value specifying the wakeup frequency parameter. It relies on the data from the *POWER_CTL* register.

void SetDataFormatSelfTestBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to SELF_TEST bit of the DATA_FORMAT register.

This function sets the SELF_TEST bit of the DATA_FORMAT register.

boolGetDataFormatSelfTestBit()

Return value
- bool – the boolean value of the SELF_TEST bit of the DATA_FORMAT register.

This function returns the SELF_TEST bit of the DATA_FORMAT **r**egister.

void SetDataFormatSPIBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to SPI bit of the DATA_FORMAT register.

This function sets the SPI bit of the DATA_FORMAT register.

boolGetDataFormatSPIBit()

Return value
- bool – the boolean value of the SPI bit of the DATA_FORMAT register.

This function returns the SPI bit of the DATA_FORMAT register.

void SetDataFormatIntInvertBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the INT_INVERT bit of the DATA_FORMAT register.

This function sets the INT_INVERT bit of the DATA_FORMAT register.

boolGetDataFormatIntInvertBit()

Return value
- bool – the boolean value of the INT_INVERT bit of the DATA_FORMAT register.

This function returns the INT_INVERT bit of the DATA_FORMAT register.

void SetDataFormatAutoSleepBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the AUTO_SLEEP bit of the DATA_FORMAT register.

This function sets the AUTO_SLEEP bit of the DATA_FORMAT register.

boolGetDataFormatAutoSleepBit()

Return value
- bool – the boolean value of the AUTO_SLEEP bit of the DATA_FORMAT register.

This function returns the AUTO_SLEEP bit of the DATA_FORMAT register.

void SetDataFormatFullResBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the FULL_RES bit of the DATA_FORMAT register.

This function sets the FULL_RES bit of the DATA_FORMAT register.

boolGetDataFormatFullResBit()

Return value
- bool – the boolean value of the FULL_RES bit of the DATA_FORMAT register.

This function returns the FULL_RES bit of the DATA_FORMAT register.

void SetDataFormatJustifyBit(bool fValue)

Parameters
- bool fValue – the boolean value to be set to the Justify bit of the DATA_FORMAT register.

This function sets the Justify bit of the DATA_FORMAT register.

boolGetDataFormatJustifyBit()

Return value
- bool – the boolean value of the Justify bit of the DATA_FORMAT register.

This function returns the Justify bit of the DATA_FORMAT register.

uint8_t SetDataFormatGRangePar(uint8_t bDataFormatGRangePar)

Parameters

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

Table 7. List of parameters for ACLSetDataFormatGRangePar function

0	PAR_DATAFORMAT_PM2G	Parameter g range : +/- 2g, used to access Range bits in the DATA_FORMAT register
1	PAR_DATAFORMAT_PM4G	Parameter g range : +/- 4g, used to access Range bits in the DATA_FORMAT register
2	PAR_DATAFORMAT_PM8G	Parameter g range : +/- 8g, used to access Range bits in the DATA_FORMAT register
3	PAR_DATAFORMAT_PM16G	Parameter g range : +/- 16g, used to access Range bits in the DATA_FORMAT register

Return value:

- ACL_ERR_SUCCESS - The action completed successfully

- ACL_ERR_ARG_RANGE_DF_03 - The argument is not within 0 - 3 range

See Errors for the errors list.

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, *ACL_ERR_ARG_RANGE_DF_03* is returned and no value is set.

uint8_t GetDataFormatGRangePar()

Return value

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

Table 8. List of parameters for ACLGetDataFormatGRangePar function

Value	Name	Note
0	PAR_DATAFORMAT_PM2G	Parameter g range : +/- 2g, used to access Range bits in DATA_FORMAT register
1	PAR_DATAFORMAT_PM4G	Parameter g range : +/- 4g, used to access Range bits in DATA_FORMAT register
2	PAR_DATAFORMAT_PM8G	Parameter g range : +/- 8g, used to access Range bits in DATA_FORMAT register
3	PAR_DATAFORMAT_PM16G	Parameter g range : +/- 16g, used to access Range bits in DATA_FORMAT register

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

void SetFIFOControlTriggerBit(bool fValue)

Parameters

- bool fValue – the boolean value to be set to the Trigger bit of the *FIFO_CTL* register.

This function sets the Trigger bit of the *FIFO_CTL* register.

boolGetFIFOControlTriggerBit()

Return value

- bool – the boolean value of the Trigger bit of the *FIFO_CTL* register.

This function returns the Trigger bit of the *FIFO_CTL* register.

uint8_t SetFIFOControlFIFOModePar(uint8_t bFIFOControlFIFOModePar)

Parameters

- uint8_t bFIFOControlFIFOModePar - the parameter specifying the FIFO mode. it can be one of the parameters in the following list:
-

Table 9. List of parameters for ACLSetFIFOControlFIFOModePar function

Value	Name	Note
0	FIFO_MODE_BYPASS	Parameter FIFO Mode : Bypass, used to access FIFO_MODE bits in FIFO_CTL register
1	FIFO_MODE_FIFO	Parameter FIFO Mode : FIFO, used to access FIFO_MODE bits in FIFO_CTL register
2	FIFO_MODE_STREAM	Parameter FIFO Mode : Stream, used to access FIFO_MODE bits in FIFO_CTL register
3	FIFO_MODE_TRIGGER	Parameter FIFO Mode : Trigger, used to access FIFO_MODE bits in FIFO_CTL register

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_FIFO_03 (8) - The argument is not within 0 - 3 range

The function sets the appropriate FIFO mode bits in the FIFO_CTL register. The accepted argument values are between 0 and 3. If the argument is within the accepted values range, it sets the FIFO mode bits in FIFO_CTL register and ACL_ERR_SUCCESS is returned.
If the value is outside this range, ACL_ERR_ARG_RANGE_FIFO_03 is returned and no value is set.

uint8_t GetFIFOControlFIFOModePar()

Return value

- uint8_t - the value specifying the FIFO mode parameter. It can be one of the values from the following list:

Table 10. List of parameters for ACLGetFIFOControlFIFOModePar function

Value	Name	Note
0	FIFO_MODE_BYPASS	Parameter FIFO Mode : Bypass, used to access FIFO_MODE bits in the FIFO_CTL register
1	FIFO_MODE_FIFO	Parameter FIFO Mode : FIFO, used to access FIFO_MODE bits in the FIFO_CTL register
2	FIFO_MODE_STREAM	Parameter FIFO Mode : Stream, used to access FIFO_MODE bits in the FIFO_CTL

		register
3	FIFO_MODE_TRIGGER	Parameter FIFO Mode : Trigger, used to access FIFO_MODE bits in the FIFO_CTL register

The function returns the value specifying the FIFO mode parameter from FIFO_CTL register.

void SetFIFOControlSamplesVal(uint8_t bFIFOControlSamplesVal)

Parameters

- uint8_t bFIFOControlSamplesVal - the value to be assigned to Samples bits in FIFO_CTL register

The function sets the appropriate Samples bits in FIFO_CTL register.

uint8_t GettFIFOControlSamplesVal()

Return value

- uint8_t - the value specifying the value of the Samples bits.

The function returns the value of the Samples bits. It relies on the data from FIFO_CTL register.

bool GetFIFOStatusFIFOTrigBit()

Return value

- bool – the boolean value of the FIFO_TRIG bit of the FIFO_STATUS register.

This function gets the FIFO_TRIG bit of the FIFO_STATUS register.

uint8_t GetFIFOStatusEntriesVal()

Return value

- uint8_t - the value of the Entries bits from FIFO_STATUS register

The function returns the Entries bits values enabled from FIFO_STATUS register.

uint8_t CalibrateOneAxisGravitational(uint8_t bAxisInfo)

Parameters

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

Table 11. List of parameters for ACLCalibrateOneAxisGravitational function

Value	Name	Axis info parameter
0	PAR_AXIS_XP	Parameter used to indicate that x axis is oriented in the gravitational direction.
1	PAR_AXIS_XN	Parameter used to indicate that x axis is oriented in the opposite gravitational direction.
2	PAR_AXIS_YP	Parameter used to indicate that y axis is oriented in the gravitational direction.

3	PAR_AXIS_YN	Parameter used to indicate that y axis is oriented in the opposite gravitational direction.
4	PAR_AXIS_ZP	Parameter used to indicate that z axis is oriented in the gravitational direction.
5	PAR_AXIS_ZN	Parameter used to indicate that z axis is oriented in the opposite gravitational direction.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully

- ACL_ERR_ARG_RANGE_05 - The argument is not within 0 - 5 range

See Errors for the errors list.

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, when function performs calibration and returns ACL_ERR_SUCCESS. If value is outside this range, ACL_ERR_ARG_RANGE_05 is returned and no calibration is done.

Interrupt related functions

uint8_t ConfigureInterrupt(uint8_t bParACLIntNo, uint8_t bParExtIntNo, uint8_t bEventMask, void (*pfIntHandler)(), uint8_t bActiveType)

Parameters

- uint8_t bParACLIntNo – The parameter indicating the ACL interrupt number. Can be one of the parameters from the following list:

Table 12. List of parameters for bParACLIntNo argument of CalibrateOneAxisGravitational function

Value	Name
0	PAR_ACL_INT1
1	PAR_ACL_INT2

- uint8_t bParExtIntNo – The parameter indicating the external interrupt number. Can be one of the parameters from the following list:

Table 13. List of parameters for bParExtIntNo argument of CalibrateOneAxisGravitational function

Value	Name
0	PAR_EXT_INT0
1	PAR_EXT_INT1
2	PAR_EXT_INT2
3	PAR_EXT_INT3
4	PAR_EXT_INT4

- uint8_t bEventMask - the events that trigger the interrupt. Can be one or more (OR-ed) parameters from the following list:

Table 14. List of values for bEventMask argument of CalibrateOneAxisGravitational function

Value	Name	corresponding bit in *INT_ENABLE & INT_MAP*
0x80	MSK_ INT_ DATA_READY	DATA_READY
0x40	MSK_ INT_ SINGLE_TAP	SINGLE_TAP
0x20	MSK_ INT_ DOUBLE_TAP	DOUBLE_TAP
0x10	MSK_ INT_ ACTIVITY	Activity
0x08	MSK_ INT_ INACTIVITY	Inactivity
0x04	MSK_ INT_ FREE_FALL	FREE_FALL
0x02	MSK_ INT_ WATERMARK	Watermark
0x01	MSK_ INT_ OVERRUN	Overrun

- void (*pfIntHandler)() - pointer to a function that will serve as interrupt handler.
- uint8_t bActiveType – The parameter indicating the interrupt pin is active high or low. Can be one of the parameters from the following list:

Table 15. List of parameters for bActiveType argument of CalibrateOneAxisGravitational function

Value	Name
0	PAR_INT_ACTIVEHIGH
1	PAR_INT_ACTIVELOW

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- a combination of the following errors(OR-ed):
- ACL_ERR_ARG_AINTNO01_BIT - ACL Interrupt no is not within 0-1 range
- ACL_ERR_ARG_EINTNO04_BIT - External Interrupt no is not within 0-4 range
- ACL_ERR_ARG_ACT_BIT - Active is different than 0 or 1

See Errors for the errors list.

The function configures the interrupt by:
- associating it to a set of events (INT_ENABLE & INT_MAP registers)
- associating it to an ACL interrupt number (1, 2)
- associating it to an external interrupt number (0-4)
- associating it to an interrupt handler
- defining the behavior for the interrupt pin.

Make sure that interrupt pin of the PmodACL corresponding to the bParACLIntNo parameter is physically connected to external interrupt pin number corresponding to bParExtIntNo parameter.
The function expects the parameters bParACLIntNo, bParExtIntNo and bActiveType to be in the specified range. For each parameter outside the range, a specific error is set and a combination (OR-ed) of the errors is returned.
If all parameters are within their range, ACL_ERR_SUCCESS is returned.

uint8_t GetInterruptSourceBits(uint8_t bEventMask)

Parameters

- uint8_t bEventMask - the events that are checked if they triggered the interrupt. Can be one or more (OR-ed) parameters from the following list.

Return value

- uint8_t - the value of the register masked using the specified mask

Table 16. List of values for bEventMask argument of GetInterruptSourceBits function

Value	Name	corresponding bit in ***INT_SOURCE***
0x80	MSK_ INT_ DATA_READY	DATA_READY
0x40	MSK_ INT_ SINGLE_TAP	SINGLE_TAP
0x20	MSK_ INT_ DOUBLE_TAP	DOUBLE_TAP
0x10	MSK_ INT_ ACTIVITY	Activity
0x08	MSK_ INT_ INACTIVITY	Inactivity
0x04	MSK_ INT_ FREE_FALL	FREE_FALL
0x02	MSK_ INT_ WATERMARK	Watermark
0x01	MSK_ INT_ OVERRUN	Overrun

The function returns the value of INT_SOURCE register masked using the provided mask.
When defining more events for an interrupt, this function can be used to determine which is causing the interrupt. Note that, according to datasheet, reading the INT_SOURCE register causes interrupt flags to be cleared.

uint8_t GetIntEnableEventBits(uint8_t bEventMask)

Parameters

- uint8_t bEventMask - the events that are checked if they are enabled to generate the interrupt. Can be one or more (OR-ed) parameters from the following list

Table 17. List of values for bEventMask argument of GetIntEnableEventBits function

Value	Name	corresponding bit in *INT_ENABLE*
0x80	MSK_ INT_ DATA_READY	DATA_READY
0x40	MSK_ INT_ SINGLE_TAP	SINGLE_TAP
0x20	MSK_ INT_ DOUBLE_TAP	DOUBLE_TAP
0x10	MSK_ INT_ ACTIVITY	Activity
0x08	MSK_ INT_ INACTIVITY	Inactivity
0x04	MSK_ INT_ FREE_FALL	FREE_FALL
0x02	MSK_ INT_ WATERMARK	Watermark
0x01	MSK_ INT_ OVERRUN	Overrun

Return value

- uint8_t - the value of the register masked using the specified mask

The function returns the value of the INT_ENABLE register masked using the provided value.

boolGetIntMapEventBits(uint8_t bParIntNo, uint8_t bEventMask)

Parameters

- uint8_t bParIntNo – The parameter indicating the interrupt number. It can be one of the parameters from the following list:

Table 18. List of parameters for bParIntNo argument of GetIntMapEventBits function

Value	Name
0	PAR_INT_INT1
1	PAR_INT_INT2

- uint8_t bEventMask - the events that are verified if mapped to generate the specific interrupt. There can be one or more (OR-ed) parameters from the following list:

Table 19. List of values for bEventMask argument of GetIntMapEventBits function

Value	Name	Bit corresponding in *INT_MAP*
0x80	MSK_ INT_ DATA_READY	DATA_READY
0x40	MSK_ INT_ SINGLE_TAP	SINGLE_TAP
0x20	MSK_ INT_ DOUBLE_TAP	DOUBLE_TAP
0x10	MSK_ INT_ ACTIVITY	Activity
0x08	MSK_ INT_ INACTIVITY	Inactivity
0x04	MSK_ INT_ FREE_FALL	FREE_FALL
0x02	MSK_ INT_ WATERMARK	Watermark
0x01	MSK_ INT_ OVERRUN	Overrun

Return value

- uint8_t - the value of the register masked using the specified value

The function returns the value of the *INT_MAP* register masked using the provided value.

uint8_t SetThresholdTapG(float dTreshTap)

Parameters

- float dTreshTap – parameter containing tap Threshold in “g”.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_016G - The argument is not within 0 - 16g range

See Errors for the errors list.

This function sets the Tap Threshold, the value being given in “g”. The accepted argument values are between 0 and 16g.
If argument is within the accepted values range, its value is quantified in 8-bit THRESH_TAP register using a scale factor of 62.5 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_016G is returned and no value is set.

float GetThresholdTapG()

Return value

- float – the value of the tap Threshold in “g”.

This function returns the Tap Threshold in “g”.

It converts the value quantified in the 8-bit THRESH_TAP register into a value expressed in “g”, using a scale factor of *62.5 mg/LSB*.

uint8_t SetThresholdActivityG(float dTreshActivity)

Parameters
- float dTreshActivity – parameter containing Activity Threshold in “g”.

Return value:
- *ACL_ERR_SUCCESS - The action completed successfully*
- *ACL_ERR_ARG_RANGE_016G - The argument is not within 0 -16g range*

See Errors for the errors list.

This function sets the Activity Threshold, the value being given in “g”. The accepted argument values are between 0 and 16g.
If argument is within the accepted values range, its value is quantified in 8-bit THRESH_ACT register using a scale factor of 62.5 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_016G is returned and no value is set.

float GetThresholdActivityG()

Return value
- float – the value of the Activity Threshold in “g”.

This function returns the Activity Threshold in “g”.
It converts the value quantified in the 8-bit THRESH_ACT register into a value expressed in “g”, using a scale factor of *62.5 mg/LSB*.

uint8_t SetThresholdInactivityG(float dTreshInactivity)

Parameters
- float dTreshInactivity – parameter containing Inactivity Threshold in “g”.

Return value:
- *ACL_ERR_SUCCESS - The action completed successfully*
- *ACL_ERR_ARG_RANGE_016G - The argument is not within 0 - 16g range*

See Errors for the errors list.

This function sets the Inactivity Threshold, the value being given in “g”. The accepted argument values are between 0 and 16g.
If argument is within the accepted values range, its value is quantified in 8-bit THRESH_INACT register using a scale factor of 62.5 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_016G is returned and no value is set.

float GetThresholdInactivityG()

Return value
- float – the value of the Inactivity Threshold in “g”.

This function returns the Inactivity Threshold in “g”.

It converts the value quantified in the 8-bit THRESH_ INACT register into a value expressed in “g”, using a scale factor of *62.5 mg/LSB*.

uint8_t SetThresholdFreeFallG(float dTreshFreeFall)

Parameters
- float dTreshFreeFall – parameter containing FreeFall Threshold in “g”.

Return value:
- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_016G - The argument is not within 0 - 16g range
See Errors for the errors list.

This function sets the FreeFall Threshold, the value being given in “g”. The accepted argument values are between 0 and 16g.
If argument is within the accepted values range, its value is quantified in 8-bit THRESH_FF register using a scale factor of 62.5 mg/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, ACL_ERR_ARG_RANGE_016G is returned and no value is set.

float GetThresholdFreeFallG()

Return value
float dTreshTap – the value of the FreeFall Threshold in “g”.

This function returns the FreeFall Threshold in “g”.
It converts the value quantified in the 8-bit THRESH_ FF register into a value expressed in “g”, using a scale factor of *62.5 mg/LSB*.

uint8_t SetDurationS(float dtDuration)

Parameters
- float dtDuration – parameter containing duration time expressed in seconds.

.

Return value:
- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_0160MS - The argument is not within 0 - 160ms range
See Errors for the errors list.

This function sets the Duration (the maximum time for which an event must be above the threshold to qualify as a tap event), the value being given in seconds. The accepted argument values are between 0 and 0.16s.
If argument is within the accepted values range, its value is quantified in 8-bit DUR register using a scale factor of 625 µs/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, *ACL_ERR_ARG_RANGE_0160MS* is returned and no value is set.
Providing a value of 0 disables the single tap/ double tap functions.

float GetDurationS()

Return value

- float dtDuration – the value of the duration time expressed in seconds.

This function returns the Duration (the maximum time for which an event must be above the threshold to qualify as a tap event) expressed in seconds.
It converts the value quantified in the 8-bit DUR register into a value expressed in seconds, using a scale factor of 625 µs/LSB.

uint8_t SetLatentS(float dtLatent)

Parameters

- float dtLatent – parameter containing Latent time expressed in seconds.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_0320MS - The argument is not within 0 - 320ms range

See Errors for the errors list.

This function sets the Latent (the wait time between the detection of a tap event and the start of the time window- defined by the window register - during which a possible second tap event can be detected), the value being given in seconds. The accepted argument values are between 0 and 320ms.
If argument is within the accepted values range, its value is quantified in 8-bit Latent register using a scale factor of 1.25 ms/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, *ACL_ERR_ARG_RANGE_0320MS* is returned and no value is set.

float GetLatentS()

Return value

float – the value of the Latent in seconds.

This function returns the Latent (the wait time between the detection of a tap event and the start of the time window - defined by the window register - during which a possible second tap event can be detected) in seconds.
It converts the value quantified in the 8-bit Latent register into a value expressed in seconds, using a scale factor of 1.25 ms/LSB.

uint8_t SetWindowS(float dtWindow)

Parameters

- float dtWindow – parameter containing Window time expressed in seconds.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_0320MS - The argument is not within 0- 320ms range

See Errors for the errors list.

This function sets the Window (the amount of time after the expiration of the latency time- determined by the latent register - during which a second valid tap can begin), the value being given in seconds. The accepted argument values are between 0 and 320ms.

If argument is within the accepted values range, its value is quantified in 8-bit Window register using a scale factor of 1.25 ms/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, *ACL_ERR_ARG_RANGE_0320MS* is returned and no value is set.
Providing a value of 0 disables the double tap function.

float GetWindowS()

Return value
float – the value of the Window time expressed in seconds

This function returns the Window (the amount of time after the expiration of the latency time-determined by the latent register - during which a second valid tap can begin) in seconds.
It converts the value quantified in the 8-bit Window register into a value expressed in seconds, using a scale factor of 1.25 ms/LSB.

uint8_t SetTimeFreeFallS(float dtTimeFreeFall)

Parameters
- float dtTimeFreeFall – parameter containing Free Fall time time expressed in seconds.

Return value:
- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_ARG_RANGE_0128S - The argument is not within 0- 1.28s range
See Errors for the errors list.

This function sets Free Fall time (the minimum time for which the value of all axes must be less than THRESH_FF in order to generate a free-fall interrupt), the value being given in seconds. The accepted argument values are between 0 and 1.28 s.
If argument is within the accepted values range, its value is quantified in 8-bit TIME_FF register using a scale factor of 5 ms/LSB and ACL_ERR_SUCCESS is returned.
If value is outside this range, *ACL_ERR_ARG_RANGE_0128S* is returned and no value is set.
Providing a value of 0 results in undesirable behavior if the free-fall interrupt is enabled. Values between 100 ms and 350 ms (0x14 to 0x46) are recommended.

float GetTimeFreeFallS()

Return value
float – the value of the Free Fall time time expressed in seconds.

This function returns the Free Fall time (the minimum time for which the value of all axes must be less than THRESH_FF in order to generate a free-fall interrupt) in seconds.
It converts the value quantified in the 8-bit TIME_FF register into a value expressed in seconds, using a scale factor of 5 ms/LSB.

void SetActiveInactiveControlBits(uint8_t bActiveInactiveControlMask, bool fValue)

Parameters
- uint8_t bActiveInactiveControlMask - the mask containing the control bits. There can be one or more (OR-ed) parameters from the following list.
- *bool fValue*
  - if true, the bits corresponding to 1 values in the mask will get 1 value

- if false, the bits corresponding to 1 values in the mask will get 0 value

Table 20. List of values for bActiveInactiveControlMask argument of SetActiveInactiveControlBits function

Value	Name	Corresponding bit in *ACT_INACT_CTL*
0x80	MSK_ACT_INACT_ACTACDC	ACT ac/dc
0x40	MSK_ACT_INACT_ACTXEN	ACT_X enable
0x20	MSK_ACT_INACT_ACTYEN	ACT_Y enable
0x10	MSK_ACT_INACT_ACTZEN	ACT_Z enable
0x08	MSK_ACT_INACT_INACTACDC	INACT ac/dc
0x04	MSK_ACT_INACT_INACTXEN	INACT_X enable
0x02	MSK_ACT_INACT_INACTYEN	INACT_Y enable
0x01	MSK_ACT_INACT_INACTZEN	INACT_Z enable

The function sets the appropriate (corresponding to the mask) bits in *ACT_INACT_CTL* register to 0 or 1 value.

uint8_t GetActiveInactiveControlBits(uint8_t bActiveInactiveControlMask)

Parameters

- uint8_t bActiveInactiveControlMask - the mask containing the control bits. There can be one or more (OR-ed) parameters from the following list.

Table 21. List of values for bActiveInactiveControlMask argument *of* GetActiveInactiveControlBits function

Value	Name	Bit corresponding in *ACT_INACT_CTL*
0x80	MSK_ACT_INACT_ACTACDC	ACT ac/dc
0x40	MSK_ACT_INACT_ACTXEN	ACT_X enable
0x20	MSK_ACT_INACT_ACTYEN	ACT_Y enable
0x10	MSK_ACT_INACT_ACTZEN	ACT_Z enable
0x08	MSK_ACT_INACT_INACTACDC	INACT ac/dc
0x04	MSK_ACT_INACT_INACTXEN	INACT_X enable
0x02	MSK_ACT_INACT_INACTYEN	INACT_Y enable
0x01	MSK_ACT_INACT_INACTZEN	INACT_Z enable

Return value

- uint8_t - the value of the register masked using the specified value

The function returns the value of *ACT_INACT_CTL* register masked using the provided value.

uint8_t GetActTapStatusBits(uint8_t bActTapStatusMask)

Parameters

- uint8_t bActTapStatusMask- the axes verified if they are the first axes involved in a tap or activity event. There can be one or more (OR-ed) parameters from the following list:

Table 22. List of values for bActTapStatusMask argument of GetActTapStatusBits function

Value	Name	Bit corresponding in *ACT_TAP_STATUS*
0x40	MSK_ACT_TAP_STATUS_ACTXSOURCE	ACT_X source
0x20	MSK_ACT_TAP_STATUS_ACTYSOURCE	ACT_Y source
0x10	MSK_ACT_TAP_STATUS_ACTZSOURCE	ACT_Z source
0x08	MSK_ACT_TAP_STATUS_ASLEEP	Asleep
0x04	MSK_ACT_TAP_STATUS_INACTXSOURCE	TAP_X source
0x02	MSK_ACT_TAP_STATUS_INACTYSOURCE	TAP_Y source
0x01	MSK_ACT_TAP_STATUS_INACTZSOURCE	TAP_Z source

Return value

- uint8_t - the value of the register masked using the specified value

The function returns the value of the *ACT_TAP_STATUS* register masked using the provided mask/value.

void SetTapAxesBits(uint8_t bTapAxesMask, bool fValue)

Parameters

- uint8_t bTapAxesMask - the mask containing the tap axes bits. There can be one or more (OR-ed) parameters from the list below:

Table 23. List of values for bTapAxesMask argument of SetTapAxesBits function

Value	Name	Bit corresponding in *TAP_AXES*
0x08	MSK_TAP_AXES_SUPPRESS	This mask is used in TAP_AXES register for Suppress bit.
0x04	MSK_TAP_AXES_TAPXENABLE	This mask is used in TAP_AXES register for TAP_X bit.
0x02	MSK_TAP_AXES_TAPYENABLE	This mask is used in TAP_AXES register for TAP_Y bit.
0x01	MSK_TAP_AXES_TAPZENABLE	This mask is used in TAP_AXES register for TAP_Z bit.

- *bool fValue*
  - if true, the bits corresponding to 1 values in the mask will get 1 value
  - if false, the bits corresponding to 1 values in the mask will get 0 value

The function sets the appropriate (corresponding to the mask) bits in *TAP_AXES* register to 0 or 1 value.

uint8_t GetActTapAxesBits(uint8_t bTapAxesMask)

Parameters

- uint8_t bTapAxesMask - the tap axes bits verified if they are enabled. There can be one or more (OR-ed) parameters from the following list:

Table 24. List of values for bTapAxesMask argument of GetActTapAxesBits function

Value	Name	Bit corresponding in *TAP_AXES*
0x08	MSK_TAP_AXES_SUPPRESS	This mask is used in TAP_AXES register for Suppress bit.
0x04	MSK_TAP_AXES_TAPXENABLE	This mask is used in TAP_AXES register for TAP_X bit.
0x02	MSK_TAP_AXES_TAPYENABLE	This mask is used in TAP_AXES register for TAP_Y bit.
0x01	MSK_TAP_AXES_TAPZENABLE	This mask is used in TAP_AXES register for TAP_Z bit.

Return value

- uint8_t - the value of the register masked using the specified value

The function returns the value of *TAP_AXES* register masked using the provided mask.

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 25. List of possible connections for PmodACL

Connection	Connector
SPI0	JB
SPI1	J1
I2C	I2C#1

Also, in order to access interrupt related functionality, an additional wire should connect the pin corresponding to PmodACL interrupt to the pin corresponding to the external interrupt used. For the Simple Demo below, the pin 8 of the PmodACL (corresponding to INT1) was connected to the pin 36 of the Cerebot 32MX4CK board (corresponding to external INT2).

- When I2C is used, make sure that PmodACL SS line is either disconnected either connected to Vcc.
- The library files have to be placed in the same location with the .pde file in a folder named **libraries** and a subfolder having the same name as the library, preferably at: C:\Users*Name*\Documents\Arduino. If the MPIDE is open, it has to be closed and reopened and under the Sketch menu ->Import library -> Contributed - the name of the library should be found.

  For example, a good way to use this library is:
  - C:\Users*Name*\Documents\Arduino\Libraries\ACL contains library files ACL.h, ACL.cpp and keywords.txt
  - C:\Users*Name*\Documents\Arduino\ ACLDemo \ contains the demo sketch ACLDemo.pde
- 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.ACLSetPowerControlMeasureBit(true);
```

Simple Demo

In order to run this demo, place the library and sketch and connect the interrupt wire as explained in the Library usage section.
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).
  - Configures the accelerometer in order to trigger interrupts on taps and double taps on Z axis, interrupts being handled by tap() function
  - Performs the PmodACL calibration
- In the loop() function:
  - Reads the accelerometer values
  - Displays the read values using the Serial interface
  - Displays information about the tap events detected by the tap() function
- In the tap() function:
  - Recognizes the interrupt source event and sets the tap information accordingly.