Digilent chipKIT ACLFULL 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 all its functionality.
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, 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

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_INVALID_THRESH	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
11	ACL_ERR_INVALID_BIT_SET	The argument is not within 0, 11 range
12	ACL_ERR_INVALID_BIT_GET	The argument is not within 0, 12 range
13	ACL_ERR_ARG_RANGE_FIFO_03	The argument is not within 0, 3 range
14	ACL_ERR_INVALID_TIME	The argument is not within 0, 3 range
16	ACL_ERR_INVALID_AXIS	The argument is not within 0, 3 range
32	ACL_ERR_ARG_AINTNO01_BIT	The argument is not within 0, 2 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

This chapter shows the ACL library functions. Read Common functions for basic ACL functions, Advanced functions and Interrupt related functions for more advanced functions.

Common 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	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 SetSFRBit(uint8_t bSFRBit, bool fValue)

Parameters

- uint8_t bSFRBit - byte indicating the SFR bit to be set. Can be one of:

Table 3. Values for bSFRBit parameter of SetSFRBit function.

	Name	Note
0	ACL_BIT_BW_RATE_LOWPOWER	LOW_POWER bit of the BW_RATE register
1	ACL_BIT_POWER_CTL_LINK	Link bit of the POWER_CTL register
2	ACL_BIT_POWER_CTL_AUTOSLEEP	AUTO_SLEEP bit of the POWER_CTL register
3	ACL_BIT_POWER_CTL_MEASURE	Measure bit of the POWER_CTL register
4	ACL_BIT_POWER_CTL_SLEEP	Sleep bit of the POWER_CTL register
5	ACL_BIT_DATA_FORMAT_SELFTEST	SELF_TEST bit of the DATA_FORMAT register
6	ACL_BIT_DATA_FORMAT_SPI	SPI bit of the DATA_FORMAT register
7	ACL_BIT_DATA_FORMAT_INTINVERT	INT_INVERT bit of the DATA_FORMAT register
8	ACL_BIT_DATA_FORMAT_AUTOSLEEP	AUTO_SLEEP bit of the DATA_FORMAT register
9	ACL_BIT_DATA_FORMAT_FULLRES	FULL_RES bit of the DATA_FORMAT register
10	ACL_BIT_DATA_FORMAT_JUSTIFY	Justify bit of the DATA_FORMAT register
11	ACL_BIT_FIFO_CTL_TRIGGER	Trigger bit of the FIFO_CTL register

- bool fValue – the boolean value to be set to the specified bit.
  - true in order to set specified bit
  - false in order to reset specified bit

Return value

- uint8_t – the error message
  - ACL_ERR_SUCCESS - The action completed successfully
  - ACL_ERR_INVALID_BIT_SET - The bSFRBit argument is not within 0 - 11 range

This function sets the specified bit to the specified value.
If the specified bit is not among the ones from the list above (bSFRBit not in 0 - 11 range), ACL_ERR_INVALID_BIT_SET is returned.

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 4. List of parameters for ACLSetDataFormatGRangePar function

Value	Name	Corresponding to:
0	PAR_DATAFORMAT_PM2G	+/- 2g
1	PAR_DATAFORMAT_PM4G	+/- 4g
2	PAR_DATAFORMAT_PM8G	+/- 8g
3	PAR_DATAFORMAT_PM16G	+/- 16g

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.

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 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

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

Advanced functions

uint8_t GetDevID()

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

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”.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- a combination (ORed) of the following errors:

- ACL_ERR_ARG_RANGE_PM2G - The dOffset argument is not within +/- 2g range
- ACL_ERR_INVALID_AXIS - The bAxisParam argument is not within 0 - 3 range

See Errors for the errors list.

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, ACL_ERR_ARG_RANGE_PM2G is returned and no value is set.
If bAxisParam parameter is outside 0 - 3 range, ACL_ERR_INVALID_AXIS is returned.

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
- uint8_t *pErr - optional parameter - pointer to a value that receives error value. User may chose not to provide this parameter. Can be used to return values:
    - ACL_ERR_SUCCESS - The action completed successfully
    - ACL_ERR_INVALID_AXIS - The argument is not within 0 - 3 range

Return value:

float – the offset for X axis in “g”.

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.
If optional parameter pErr is provided, it will point to the error value: ACL_ERR_SUCCESS if success or ACL_ERR_INVALID_AXIS if bAxisParam is not within 0 - 3 range.

bool GetSFRBit(uint8_t bSFRBit, uint8_t *pErr)

Parameters

- uint8_t bSFRBit - byte indicating the SFR bit to be set. Can be one of:

Table 5. Values for bSFRBit parameter of GetSFRBit function.

	Name	Note
0	ACL_BIT_BW_RATE_LOWPOWER	LOW_POWER bit of the BW_RATE register
1	ACL_BIT_POWER_CTL_LINK	Link bit of the POWER_CTL register
2	ACL_BIT_POWER_CTL_AUTOSLEEP	AUTO_SLEEP bit of the POWER_CTL register
3	ACL_BIT_POWER_CTL_MEASURE	Measure bit of the POWER_CTL register
4	ACL_BIT_POWER_CTL_SLEEP	Sleep bit of the POWER_CTL register
5	ACL_BIT_DATA_FORMAT_SELFTEST	SELF_TEST bit of the DATA_FORMAT register
6	ACL_BIT_DATA_FORMAT_SPI	SPI bit of the DATA_FORMAT register
7	ACL_BIT_DATA_FORMAT_INTINVERT	INT_INVERT bit of the DATA_FORMAT register

8	ACL_BIT_DATA_FORMAT_AUTOSLEEP	AUTO_SLEEP bit of the DATA_FORMAT register
9	ACL_BIT_DATA_FORMAT_FULLRES	FULL_RES bit of the DATA_FORMAT register
10	ACL_BIT_DATA_FORMAT_JUSTIFY	Justify bit of the DATA_FORMAT register
11	ACL_BIT_FIFO_CTL_TRIGGER	Trigger bit of the FIFO_CTL register
12	ACL_BIT_FIFO_STATUS_FIFOTRIG	FIFO_TRIG bit of the FIFO_STATUS register

- uint8_t *pErr - optional parameter - pointer to a value that receives error value. User may chose not to provide this parameter. Can be used to return values:
  - ACL_ERR_SUCCESS - The action completed successfully
  - ACL_ERR_INVALID_BIT_GET - The bSFRBit argument is not within 0 - 12 range

Return value

- bool the boolean value of the specified bit.

This function returns the boolean value of the specified bit.
If optional parameter pErr is provided, it will point to the error value: ACL_ERR_SUCCESS if success or ACL_ERR_INVALID_BIT_GET if bSFRBit is not within 0 - 12 range.

uint8_t SetBWRateOutputRatePar(uint8_t bOutputRatePar)

Parameters

- uint8_t bOutputRatePar – a parameter selecting one of the options from the following list.

Table 6 List of values for bOutputRatePar parameter in SetBWRateOutputRatePar function

Value	Name	Corresponding to
0	PAR_OUTPUTRATE0_10Hz	0.10 Hz
1	PAR_OUTPUTRATE0_20Hz	0.20 Hz
2	PAR_OUTPUTRATE0_39Hz	0.39 Hz
3	PAR_OUTPUTRATE0_78Hz	0.78 Hz
4	PAR_OUTPUTRATE1_56Hz	1.56 Hz
5	PAR_OUTPUTRATE3_13Hz	3.13 Hz
6	PAR_OUTPUTRATE6_25Hz	6.25 Hz
7	PAR_OUTPUTRATE12_50Hz	12.5 Hz
8	PAR_OUTPUTRATE25_00Hz	25 Hz
9	PAR_OUTPUTRATE50_00Hz	50 Hz
10	PAR_OUTPUTRATE100_00Hz	100 Hz
11	PAR_OUTPUTRATE200_00Hz	200 Hz
12	PAR_OUTPUTRATE400_00Hz	400 Hz
13	PAR_OUTPUTRATE800_00Hz	800 Hz
14	PAR_OUTPUTRATE1600_00Hz	1600 Hz
15	PAR_OUTPUTRATE3200_00Hz	3200 Hz

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 7. List of values returned by GetBWRateOutputRatePar function

Value	Name	Corresponding to
0	PAR_OUTPUTRATE0_10Hz	0.10 Hz
1	PAR_OUTPUTRATE0_20Hz	0.20 Hz
2	PAR_OUTPUTRATE0_39Hz	0.39 Hz
3	PAR_OUTPUTRATE0_78Hz	0.78 Hz
4	PAR_OUTPUTRATE1_56Hz	1.56 Hz
5	PAR_OUTPUTRATE3_13Hz	3.13 Hz
6	PAR_OUTPUTRATE6_25Hz	6.25 Hz
7	PAR_OUTPUTRATE12_50Hz	12.5 Hz
8	PAR_OUTPUTRATE25_00Hz	25 Hz
9	PAR_OUTPUTRATE50_00Hz	50 Hz
10	PAR_OUTPUTRATE100_00Hz	100 Hz
11	PAR_OUTPUTRATE200_00Hz	200 Hz
12	PAR_OUTPUTRATE400_00Hz	400 Hz
13	PAR_OUTPUTRATE800_00Hz	800 Hz
14	PAR_OUTPUTRATE1600_00Hz	1600 Hz
15	PAR_OUTPUTRATE3200_00Hz	3200 Hz

This function returns a value corresponding to the Output Rate bits in the BW_RATE 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 8. List of values for bOutputRatePar parameter in SetBWRateOutputRatePar function ACLSetPowerControlWakeupFreqPar function

Value	Name	Corresponding to
0	PAR_WAKEUP_FREQ8Hz	8 Hz
1	PAR_WAKEUP_FREQ4Hz	4 Hz
2	PAR_WAKEUP_FREQ2Hz	2 Hz
3	PAR_WAKEUP_FREQ1Hz	1 Hz

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 9. List of values for ACLGetPowerControlWakeupFreqPar function

Value	Name	Corresponding to
0	PAR_WAKEUP_FREQ8Hz	8 Hz
1	PAR_WAKEUP_FREQ4Hz	4 Hz
2	PAR_WAKEUP_FREQ2Hz	2 Hz
3	PAR_WAKEUP_FREQ1Hz	1 Hz

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

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 10. List of parameters for ACLGetDataFormatGRangePar function

Value	Name	Corresponding to:
0	PAR_DATAFORMAT_PM2G	+/- 2g
1	PAR_DATAFORMAT_PM4G	+/- 4g
2	PAR_DATAFORMAT_PM8G	+/- 8g
3	PAR_DATAFORMAT_PM16G	+/- 16g

The function returns the value specifying the g range parameter. It relies on the Range bits in DATA_FORMAT 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 11. List of parameters for ACLSetFIFOControlFIFOModePar function

Value	Name	FIFO Mode
0	FIFO_MODE_BYPASS	Bypass
1	FIFO_MODE_FIFO	FIFO

2	FIFO_MODE_STREAM	Stream
3	FIFO_MODE_TRIGGER	Trigger

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 12. List of parameters for ACLGetFIFOControlFIFOModePar function

Value	Name	FIFO Mode
0	FIFO_MODE_BYPASS	Bypass
1	FIFO_MODE_FIFO	FIFO
2	FIFO_MODE_STREAM	Stream
3	FIFO_MODE_TRIGGER	Trigger

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.

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 13. 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.

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 14. 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 15. 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 16. 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 17. 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 (by setting 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 18. 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 19. 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.

bool GetIntMapEventBits(uint8_t bParIntNo, uint8_t bEventMask , uint8_t *pErr)

Parameters

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

Table 20. 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 21. 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

- uint8_t *pErr - optional parameter - pointer to a value that receives error value. User may chose not to provide this parameter. Can be used to return values:
  - ACL_ERR_SUCCESS - The action completed successfully
  - ACL_ERR_ARG_AINTNO02_BIT - bParIntNo is not within 0-1 range

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.
If optional parameter pErr is provided, it will point to the error value: ACL_ERR_SUCCESS if success or ACL_ERR_ARG_AINTNO02_BIT if bParIntNo (ACL Interrupt no) is not within 0-1 range.

uint8_t SetThresholdG(uint8_t bThreshParam, float dTreshVal)

Parameters

- uint8_t bThreshParam - byte indicating the threshold that will be set. Can be one of:

Table 22. Values for bThreshParam of SetThresholdG function

Value	Name	Corresponding to
0	PAR_THRESH_TAP	Threshold Tap
1	PAR_THRESH_ACT	Threshold Activity
2	PAR_THRESH_INACT	Threshold Inactivity
3	PAR_THRESH_FF	Threshold Free Fall

- float dTreshVal – parameter containing Threshold value in “g”.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- a combination (ORed) of the following errors:
  - ACL_ERR_ARG_RANGE_016G - The dTreshVal argument is not within 0 - 16g range
  - ACL_ERR_INVALID_THRESH - The bThreshParam argument is not within 0 - 3 range

See Errors for the errors list.

This function sets the Tap, Activity, Inactivity or Free Fall Threshold, the value being given in “g”. The bThreshParam parameter specify which of the threshold values will be set.
The accepted threshold values are between 0 and 16g. If value is outside this range, ACL_ERR_ARG_RANGE_016G is returned and no value is set.
If argument is within the accepted values range, its value is quantified in 8-bit threshold register using a scale factor of 62.5 mg/LSB and ACL_ERR_SUCCESS is returned.

float GetThresholdG(uint8_t bThreshParam, uint8_t *pErr)

Parameters

- uint8_t bThreshParam - byte indicating the threshold that will be set. Can be one of:

Table 23. Values for bThreshParam of GetThresholdG function

Value	Name	Corresponding to
0	PAR_THRESH_TAP	Threshold Tap
1	PAR_THRESH_ACT	Threshold Activity
2	PAR_THRESH_INACT	Threshold Inactivity
3	PAR_THRESH_FF	Threshold Free Fall

- uint8_t *pErr - optional parameter - pointer to a value that receives error value. User may chose not to provide this parameter. Can be used to return values:
  - ACL_ERR_SUCCESS - The action completed successfully
  - ACL_ERR_INVALID_THRESH - The bThreshParam argument is not within 0 - 3 range

Return value

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

This function returns the Tap, Activity, Inactivity or Free Fall Threshold, in “g”. The bThreshParam parameter specify which of the threshold values will be returned.
It converts the 8-bit value quantified in the specific threshold register into a value expressed in “g”, using a scale factor of 62.5 mg/LSB.
If optional parameter pErr is provided, it will point to the error value: ACL_ERR_SUCCESS if success or ACL_ERR_INVALID_THRESH if bThreshParam is not within 0 - 3 range.

uint8_t SetTimeS(uint8_t bTimeParam, float dTime)

Parameters

- uint8_t bTimeParam - byte indicating the Time that will be set. Can be one of:

Table 24. Values for bTimeParam of SetTimeG function

	Name	to	Range	Error	Scale factor
0	PAR_TIME_DUR	Duration	0 - 0.16s	ACL_ERR_ARG_RANGE_0160MS	625 µs/LSB
1	PAR_TIME_LATENT	Latent	0 - 0.32s	ACL_ERR_ARG_RANGE_DF_0320MS	1.25 ms/LSB
2	PAR_TIME_WINDOW	Window	0 - 0.32s	ACL_ERR_ARG_RANGE_DF_0320MS	1.25 ms/LSB
3	PAR_TIME_FF	Free Fall	0 - 1.28s	ACL_ERR_ARG_RANGE_0128S	5 ms/LSB

- float dTime - parameter containing time expressed in seconds.

Return value:

- ACL_ERR_SUCCESS - The action completed successfully
- ACL_ERR_INVALID_TIME - The bTimeParam is not in 0 - 3 range
- Depending on bTimeParam (see table), one of:
  - ACL_ERR_ARG_RANGE_0160MS - The argument is not within 0 - 160ms range
  - ACL_ERR_ARG_RANGE_DF_0320MS - The argument is not within 0 - 320ms range
  - ACL_ERR_ARG_RANGE_0128S - The argument is not within 0- 1.28s range

See Errors for the errors list.

This function sets the Duration, Latent, Window and Free Fall time, the value being given in seconds. The bTimeParam parameter specify which of the time values will be set. See the table above for the accepted range values and for the errors in case they are not respected.
If argument is within the accepted values range, its value is quantified in specific 8-bit register using the scale factor described in the table and ACL_ERR_SUCCESS is returned.

float GetTimeG(uint8_t bTimeParam, uint8_t *pErr)

Parameters

- uint8_t bTimeParam - byte indicating the Time that will be set. Can be one of:

Table 25. Values for bTimeParam of GetTimeG function

Val.	Name	to	Error	Scale factor
0	PAR_TIME_DUR	Duration	ACL_ERR_ARG_RANGE_0160MS	625 µs/LSB
1	PAR_TIME_LATENT	Latent	ACL_ERR_ARG_RANGE_DF_0320MS	1.25 ms/LSB
2	PAR_TIME_WINDOW	Window	ACL_ERR_ARG_RANGE_DF_0320MS	1.25 ms/LSB
3	PAR_TIME_FF	Free Fall	ACL_ERR_ARG_RANGE_0128S	5 ms/LSB

- uint8_t *pErr - optional parameter - pointer to a value that receives error value. User may chose not to provide this parameter. Can be used to return values:
  - ACL_ERR_SUCCESS - The action completed successfully
  - ACL_ERR_INVALID_TIME - The bTimeParam argument is not within 0 - 3 range

Return value

- float – the value of the specified time expressed in seconds.

This function returns the Duration, Latent, Window and Free Fall time expressed in seconds. The bTimeParam parameter specify which of the time values will be set.
It converts the value quantified in the 8-bit register into a value expressed in seconds, using the scale factor from the above table.
If optional parameter pErr is provided, it will point to the error value: ACL_ERR_SUCCESS if success or ACL_ERR_INVALID_TIME if bTimeParam is not within 0 - 3 range.

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 26. 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 27. 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 28. 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 29. List of values for bTapAxesMask argument of **SetTapAxesBits function**

Value	Name	Bit corresponding in *TAP_AXES*
0x08	MSK_TAP_AXES_SUPPRESS	Suppress
0x04	MSK_TAP_AXES_TAPXENABLE	TAP_X
0x02	MSK_TAP_AXES_TAPYENABLE	TAP_Y
0x01	MSK_TAP_AXES_TAPZENABLE	TAP_Z

- *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 30. List of values for bTapAxesMask argument of **GetActTapAxesBits function**

Value	Name	Bit corresponding in *TAP_AXES*
0x08	MSK_TAP_AXES_SUPPRESS	Suppress
0x04	MSK_TAP_AXES_TAPXENABLE	TAP_X
0x02	MSK_TAP_AXES_TAPYENABLE	TAP_Y
0x01	MSK_TAP_AXES_TAPZENABLE	TAP_Z

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 31. 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.
- 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.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.