python-apds9930 Documentation

Release 0.1

Davide Depau

Sep 12, 2017

Contents

This module allows you to interface with an Avago APDS-9930 I2C ambient light and proximity sensor from Python.

The bindings are easy to use: just create an instance of the APDS9930 class passing the bus number as a parameter, and optionally the device's I2C address, if it differs from the default (0x39, 57).

Almost all features are provided through properties, which can just be read or set to retrieve or send the value to/from the device.

Methods are provided to do some more complex operations that require a minimum pre-initialization: *enable_proximity_sensor()* and *enable_ambient_light_sensor()*. Those two sensors can also be enabled manually by setting the required settings first and then setting *power* and *proximity_sensor* or *ambient_light_sensor* to True to turn them on.

CHAPTER 1

Classes

class apds9930.**APDS9930**(*bus*, *address=57*)

Bases: *apds9930.APDS9930_I2C_Base*

The APDS-9930 I2C interface class. Pass the `bus` number and, if it's different from 0x39 (57), the device `address` as arguments.

The device is initialized with the default settings. bus must be an int curresponding to the number of the I2C bus you want to use. `address` is optional and is the I2C address of the device, if different from the default (0x39).

ambient_light

Ambient light value in lux (read-only).

ambient_light_gain

Receiver gain for ambient light sensor. Good values are:

Value	Gain	*apds9930.values* constant name
0	1x	`AGAIN_1X`
1	4x	`AGAIN_8X`
2	16x	`AGAIN_16X`
3	64x	`AGAIN_120X`

ambient_light_int_high_threshold

Ambient light interrupt high threshold.

ambient_light_int_low_threshold

Ambient light interrupt low threshold.

ambient_light_interrupt

If True, the device is asserting an ambient light interrupt. Set it to None to clear it.

ambient_light_sensor

Enable or disable the ambient light sensor (mode, boolean).

ambient_to_lux(*ch0*, *ch1*)

Accepts data from both channels and returns a value in lux (according to the datasheet).

ch0_light
Light data from channel 0. Read-only.

ch1_light
Light data from channel 1. Read-only.

clear_all_interrupts()
Clear all interrupts.

dump_registers()
Debug: read all the registers from the device and **print** them.

enable_ambient_light_interrupt
Enable or disable the ambient light interrupt (mode, boolean).

enable_ambient_light_sensor(*interrupt=False*)
Set all the needed values to turn on the ambient light sensor and turn it on. If interrupt is True, ALS interrupts will also be enabled.

enable_proximity_interrupt
Enable or disable the proximity interrupt (mode, boolean).

enable_proximity_sensor(*interrupt=False*)
Set all the needed values to turn on the proximity sensor and turn it on. If interrupt is True, proximity interrupts will also be enabled.

get_mode(*mode*)
Gets the state of a specific feature in the ENABLE register. Good values for mode are:

Mode	#
POWER	0
AMBIENT_LIGHT	1
PROXIMITY	2
WAIT	3
AMBIENT_LIGHT_INT	4
PROXIMITY_INT	5
SLEEP_AFTER_INT	6

The specified feature is either enabled or disabled depending on whether the method returns True or False

id
The ID of the device, stored in the ID register.

led_drive
LED drive strength for proximity and ALS. Good values are:

Value	LED Current	*apds9930.values* constant name
0	100 mA	LED_DRIVE_100MA
1	50 mA	LED_DRIVE_50MA
2	25 mA	LED_DRIVE_25MA
3	12.5 mA	LED_DRIVE_12_5MA

mode
The value of the ENABLE register, which stores the enabled features of the sensor. You should not set this property directly unless you know what you're doing. Use *set_mode()* and/or the specific feature methods instead.

power
Turn on or off the internal oscillator (mode, boolean).

proximity
Proximity data. Read-only.

proximity_diode
Diode used for proximity sensor. Good values are:

Value	Diode selection
0	Reserved
1	Reserved
2	Use Ch1 diode
3	Reserved

proximity_gain
Receiver gain for proximity detection. Good values are:

Value	Gain	*apds9930.values* constant name
0	1x	PGAIN_1X
1	2x	PGAIN_2X
2	4x	PGAIN_4X
3	8x	PGAIN_8X

proximity_int_high_threshold
Proximity interrupt high threshold.

proximity_int_low_threshold
Proximity interrupt low threshold.

proximity_interrupt
If True, the device is asserting a proximity interrupt. Set it to None to clear it.

proximity_sensor
Enable or disable the proximity sensor (mode, boolean).

set_mode(*mode*, *enable*)
Like *get_mode()*, but changes the mode instead. The enable argument determines whether the feature specified by mode will be enabled or disabled. The method accepts one additional argument as *mode*,

Mode	#
ALL	7

which enables or disables all features at once.

The specified feature will either be enabled or disabled depending on whether enable is True or False

sleep_after_interrupt
Enable or disable the sleep after interrupt feature. If True, the device will power down after an interrupt has been generated (mode, boolean).

wait_timer
Enable or disable the wait timer feature (mode, boolean).

class apds9930.**APDS9930_I2C_Base**(*bus*, *address*)
Base class for APDS9930 that provides basic I2C communication methods, specifically adapted for this device.

bus must be an integer curresponding to the I2C bus you want to use. address is the I2C address of the device.

close()
Close the I2C bus.

read_block_data(*reg*, *len*, *mode=160*)
Read a block with size len starting from the specified address.

read_byte()
Read a byte from the I2C bus.

read_byte_data(*reg*, *mode=160*)
Read a byte from the specified address.

write_block_data(*reg*, *data*, *mode=160*)
Write a block starting from a specific register. Mode should be set to AUTO_INCREMENT, so that the device automatically selects the following register before writing the next byte.

write_byte(*data*)
Write a byte to the specified address. Useful to interact with the COMMAND register directly.

write_byte_data(*reg*, *data*, *mode=160*)
Write a byte to a specific register. mode can be found in apds9930.values, and can be AUTO_INCREMENT, REPEATED_BYTE or SPECIAL_FN. Check the device's datasheet for more information.

exception apds9930.**SensorError**
Bases: exceptions.EnvironmentError

Raised when a non-I2C-specific error occurs (for example, the device ID is not recognized, which usually means the device is not hooked up properly).

If issues with the I2C occur, an IOError will be raised.

CHAPTER 2

Constants

Constants and default values used by the library

These values are defined in the *apds9930.values* module.

COMMAND register modes (see `write_byte()`)

REPEATED_BYTE = 0x80

AUTO_INCREMENT = 0xA0

SPECIAL_FN = 0xE0

APDS-9930 I2C address

APDS9930_I2C_ADDR = 0x39

Acceptable device IDs

APDS9930_IDs = [0x39]

APDS-9930 register addresses

APDS9930_ENABLE = 0x00

APDS9930_ATIME = 0x01

APDS9930_WTIME = 0x03

APDS9930_AILTL = 0x04

APDS9930_AILTH = 0x05

APDS9930_AIHTL = 0x06

APDS9930_AIHTH = 0x07

APDS9930_PILTL = 0x08

APDS9930_PILTH = 0x09

APDS9930_PIHTL = 0x0A

APDS9930_PIHTH = 0x0B

APDS9930_PERS = 0x0C

APDS9930_CONFIG = 0x0D

APDS9930_PPULSE = 0x0E

APDS9930_CONTROL = 0x0F

APDS9930_ID = 0x12

APDS9930_STATUS = 0x13

APDS9930_Ch0DATAL = 0x14

APDS9930_Ch0DATAH = 0x15

APDS9930_Ch1DATAL = 0x16

APDS9930_Ch1DATAH = 0x17

APDS9930_PDATAL = 0x18

APDS9930_PDATAH = 0x19

APDS9930_POFFSET = 0x1E

List for printing purposes and for iteration

REGISTERS = {registers}

Bit fields

APDS9930_PON = 0b00000001

APDS9930_AEN = 0b00000010

APDS9930_PEN = 0b00000100

APDS9930_WEN = 0b00001000

APSD9930_AIEN = 0b00010000

APDS9930_PIEN = 0b00100000

APDS9930_SAI = 0b01000000

On/Off definitions

OFF = 0

ON = 1

Acceptable parameters for `set_mode()`

POWER = 0

AMBIENT_LIGHT = 1

PROXIMITY = 2

WAIT = 3

AMBIENT_LIGHT_INT = 4

PROXIMITY_INT = 5

SLEEP_AFTER_INT = 6

ALL = 7

LED Drive values (`led_drive`)

LED_DRIVE_100MA = 0

LED_DRIVE_50MA = 1

LED_DRIVE_25MA = 2

LED_DRIVE_12_5MA = 3

Proximity Gain (PGAIN) values (`proximity_gain`)

PGAIN_1X = 0

PGAIN_2X = 1

PGAIN_4X = 2

PGAIN_8X = 3

ALS Gain (AGAIN) values (`ambient_light_gain`)

AGAIN_1X = 0

AGAIN_8X = 1

AGAIN_16X = 2

AGAIN_120X = 3

Interrupt clear values

CLEAR_PROX_INT = 0xE5

CLEAR_ALS_INT = 0xE6

CLEAR_ALL_INTS = 0xE7

Default values

DEFAULT_ATIME = 0xFF

DEFAULT_WTIME = 0xFF

DEFAULT_PTIME = 0xFF

DEFAULT_PPULSE = 0x08

DEFAULT_POFFSET = 0

DEFAULT_CONFIG = 0

DEFAULT_PDRIVE = LED_DRIVE_100MA

DEFAULT_PDIODE = 2

DEFAULT_PGAIN = PGAIN_8X

DEFAULT_AGAIN = AGAIN_16X

DEFAULT_PILT = 0

DEFAULT_PIHT = 50

DEFAULT_AILT = 0xFFFF

DEFAULT_AIHT = 0

DEFAULT_PERS = 0x22

ALS coefficients

DF = 52

GA = 0.49

B = 1.862

C = 0.746

D = 1.291

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index