

PALS-2

The documentation for the Arduino Library

Yuxi Sun

About this document

Contents

1	Arduino Library for the proximity and ambient light sensor PALS-2	3
1.1	Introduction	3
2	Bug List	4
3	Data Structure Documentation	5
3.1	Pals2 Class Reference	5
3.1.1	Member Function Documentation	7
3.1.1.1	begin()	7
3.1.1.2	disableAmbientLightInterrupt()	7
3.1.1.3	disableProximityInterrupt()	7
3.1.1.4	disableProximityOffsetCompensation()	7
3.1.1.5	enableAmbientLightInterrupt()	8
3.1.1.6	enableColorCompensation()	8
3.1.1.7	enablePeriodicMeasurements()	8
3.1.1.8	enableProximityInterrupt()	8
3.1.1.9	enableProximityOffsetCompensation()	9
3.1.1.10	getIlluminance()	9
3.1.1.11	getRawAmbientLight()	9
3.1.1.12	getRawAmbientLightOnDemand()	9
3.1.1.13	getRawProximity()	9
3.1.1.14	getRawProximityOnDemand()	10
3.1.1.15	setADCGain()	10
3.1.1.16	setAmbientLightMeasurementRate()	10
3.1.1.17	setInterruptPersistence()	10
3.1.1.18	setProximityMeasurementRate()	11
3.1.1.19	updateData()	11
	Index	12

1 Arduino Library for the proximity and ambient light sensor PALS-2

1.1 Introduction

The infineon PALS-2 (packaged by Vishay as VCNL4135X01) is a proximity and ambient light sensor. It offers proximity and ambient light readings with 16-bit resolution. I2C protocol is used to communicate with the host microcontroller. It can be used for gesture recognition, touch screen locking and dimming of displays.

For the proximity function there are a built-in IRED driver and photo-pin-diode. LED driver current can be programmed and up to 3 external IREDs can be connected. Offset compensation can be enabled for the proximity measurement; with this feature the sensor writes the difference between the normal proximity value and the estimated offset into the corresponding register.

For the ambient light function there a one photo-pin-diode. Two additional photodiodes can receive light in the blue area.

Other features include: readouts either periodically or on-demand; interrupts for both functions, with adjustable lower/upper thresholds and persistence.

2 Bug List

File [Pals2.cpp](#)

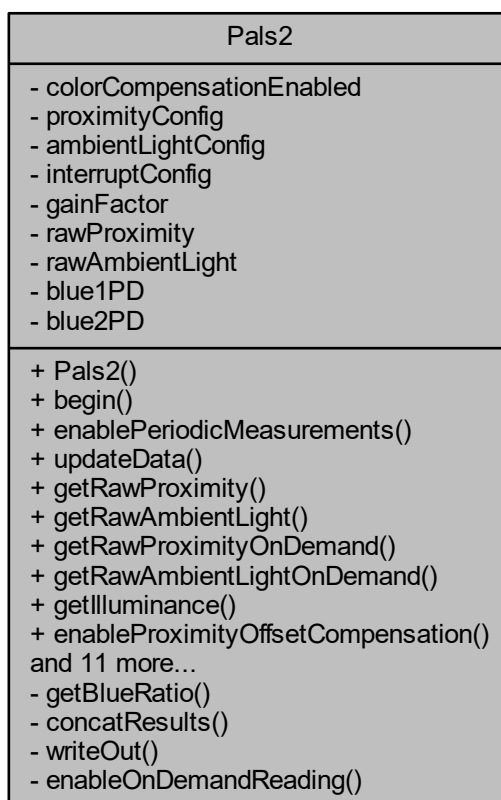
no Blue-PD value updates -> getIlluminance() not working;

in register 83h sensor measurement freezes if IRED output is not default(0): due to circuitry?

3 Data Structure Documentation

3.1 Pals2 Class Reference

Collaboration diagram for Pals2:



Public Member Functions

- void [begin](#) (void)
Starts the sensor.
- void [enablePeriodicMeasurements](#) (void)
Enables periodic measurements of proximity and ambient light values.
- void [updateData](#) (void)
Updates measurement data. Needed to be called in each measurement cycle.
- uint16_t [getRawProximity](#) (void)
Gets sensor measurement updates. Should be called after [updateData\(void\)](#) [updateData\(\)](#).

- uint16_t **getRawAmbientLight** (void)
Gets sensor measurement updates. Should be called after `updateData(void)` `updateData()`
- uint16_t **getRawProximityOnDemand** (void)
- uint16_t **getRawAmbientLightOnDemand** (void)
- float **getIlluminance** (void)
- void **enableProximityOffsetCompensation** (void)
Enables proximity offset compensation. The raw proximity values read will be the difference between the actual measured value and the estimated offset value, thus 2 measurements are taken in each cycle. Works for both periodic and on-demand measurement.
- void **disableProximityOffsetCompensation** (void)
Disables proximity offset compensation.
- void **setProximityMeasurementRate** (uint16_t rate)
Sets the measurement rate of proximity measurement.
- void **setInterruptPersistence** (uint8_t persistence)
Sets the number of consecutive measurements needed above/below the threshold for an interrupt to be generated.
- void **enableProximityInterrupt** (uint16_t topThreshold=0xFF, uint16_t bottomThreshold=0x00)
Enables interrupts for proximity measurement and sets the lower/upper thresholds.
- void **disableProximityInterrupt** (void)
Disables interrupts for proximity measurement.
- void **enableAmbientLightInterrupt** (uint16_t topThreshold=0xFF, uint16_t bottomThreshold=0x00)
Enables interrupts for ambient light measurement and sets the lower/upper thresholds.
- void **disableAmbientLightInterrupt** (void)
Disables interrupts for ambient light measurement.
- void **enableColorCompensation** (bool colorCompPeriod=0)
For light sources with high intensity color compensation should be enabled (additional reading of blue PD will be conducted).
- void **setADCGain** (uint16_t adcGain)
Sets the ADC gain, which affects the calculation of illuminance. A higher ADC gain leads to a higher illuminance value.
- void **setAmbientLightMeasurementRate** (uint8_t alsRate)
Sets the rate of ambient light measurement. Number of measurements per second, which is an integer from 1 to 8.
- void **resetSensor** (void)

Private Member Functions

- float **getBlueRatio** (void)
- uint16_t **concatResults** (uint8_t upperByte, uint8_t lowerByte)
- void **writeOut** (uint16_t regNum, uint16_t val)
- void **enableOnDemandReading** (void)

Private Attributes

- bool **colorCompensationEnabled** = false

- `uint8_t proximityConfig = 0`
- `uint8_t ambientLightConfig = 0`
- `uint8_t interruptConfig = 0`
- `float gainFactor = 81.79`
- `uint16_t rawProximity = 0`
- `uint16_t rawAmbientLight = 0`
- `uint16_t blue1PD = 0`
- `uint16_t blue2PD = 0`

3.1.1 Member Function Documentation

3.1.1.1 `begin()`

```
void Pals2::begin (  
    void )
```

Starts the sensor.

3.1.1.2 `disableAmbientLightInterrupt()`

```
void Pals2::disableAmbientLightInterrupt (  
    void )
```

Disables interrupts for ambient light measurement.

3.1.1.3 `disableProximityInterrupt()`

```
void Pals2::disableProximityInterrupt (  
    void )
```

Disables interrupts for proximity measurement.

3.1.1.4 `disableProximityOffsetCompensation()`

```
void Pals2::disableProximityOffsetCompensation (  
    void )
```

Disables proximity offset compensation.

3.1.1.5 enableAmbientLightInterrupt()

```
void Pals2::enableAmbientLightInterrupt (
    uint16_t topThreshold = 0xFF,
    uint16_t bottomThreshold = 0x00 )
```

Enables interrupts for ambient light measurement and sets the lower/upper thresholds.

Parameters

<i>topThreshold</i>	Upper threshold. By default 65536
<i>bottomThreshold</i>	Lower threshold. By default 0

3.1.1.6 enableColorCompensation()

```
void Pals2::enableColorCompensation (
    bool colorCompPeriod = 0 )
```

For light sources with high intensity color compensation should be enabled (additional reading of blue PD will be conducted).

Parameters

<i>colorCompPeriod</i>	The period of color compensation measurement; 0 for a shorter period (0 to 10ms) and 1 for a longer period (10 to 100ms)
------------------------	--------------------------------------------------------------------------------------------------------------------------

3.1.1.7 enablePeriodicMeasurements()

```
void Pals2::enablePeriodicMeasurements (
    void )
```

Enables periodic measurements of proximity and ambient light values.

3.1.1.8 enableProximityInterrupt()

```
void Pals2::enableProximityInterrupt (
    uint16_t topThreshold = 0xFF,
    uint16_t bottomThreshold = 0x00 )
```

Enables interrupts for proximity measurement and sets the lower/upper thresholds.

Parameters

<i>topThreshold</i>	Upper threshold. By default 65536
<i>bottomThreshold</i>	Lower threshold. By default 0

3.1.1.9 enableProximityOffsetCompensation()

```
void Pals2::enableProximityOffsetCompensation (
    void )
```

Enables proximity offset compensation. The raw proximity values read will be the difference between the actual measured value and the estimated offset value, thus 2 measurements are taken in each cycle. Works for both periodic and on-demand measurement.

3.1.1.10 getIlluminance()

```
float Pals2::getIlluminance (
    void )
```

Returns

the illuminance value computed from ALS and blue photodiode values.

3.1.1.11 getRawAmbientLight()

```
uint16_t Pals2::getRawAmbientLight (
    void )
```

Gets sensor measurement updates. Should be called after [updateData\(void\)](#) [updateData\(\)](#)

Returns

raw ambient light value as an integer from 0 to 65536.

3.1.1.12 getRawAmbientLightOnDemand()

```
uint16_t Pals2::getRawAmbientLightOnDemand (
    void )
```

Returns

a single raw ambient light value measured on demand.

3.1.1.13 getRawProximity()

```
uint16_t Pals2::getRawProximity (
    void )
```

Gets sensor measurement updates. Should be called after [updateData\(void\)](#) [updateData\(\)](#).

Returns

raw proximity value as an integer from 0 to 65536.

3.1.1.14 getRawProximityOnDemand()

```
uint16_t Pals2::getRawProximityOnDemand (
    void )
```

Returns

a single raw proximity value measured on demand.

3.1.1.15 setADCGain()

```
void Pals2::setADCGain (
    uint16_t adcGain )
```

Sets the ADC gain, which affects the calculation of illuminance. A higher ADC gain leads to a higher illuminance value.

Parameters

<i>adcGain</i>	ADC gain in fA, can be 200/800/3200/25600; for any other value the default (200 fA) is taken
----------------	----------------------------------------------------------------------------------------------

3.1.1.16 setAmbientLightMeasurementRate()

```
void Pals2::setAmbientLightMeasurementRate (
    uint8_t alsRate )
```

Sets the rate of ambient light measurement. Number of measurements per second, which is an integer from 1 to 8.

3.1.1.17 setInterruptPersistence()

```
void Pals2::setInterruptPersistence (
    uint8_t persistence )
```

Sets the number of consecutive measurements needed above/below the threshold for an interrupt to be generated.

Parameters

<i>persistence</i>	Number of valid measurements needed, which is one of the numbers from [1, 2, 4, 8, 16, 32, 64, 128].
--------------------	------------------------------------------------------------------------------------------------------

3.1.1.18 setProximityMeasurementRate()

```
void Pals2::setProximityMeasurementRate (
    uint16_t rate )
```

Sets the measurement rate of proximity measurement.

Parameters

<i>rate</i>	Number of measurements per second. Can be one of the numbers from [2, 4, 8, 16, 32, 64, 128, 256].
-------------	----------------------------------------------------------------------------------------------------

3.1.1.19 updateData()

```
void Pals2::updateData (
    void )
```

Updates measurement data. Needed to be called in each measurement cycle.

Index

begin
 Pals2, [7](#)

disableAmbientLightInterrupt
 Pals2, [7](#)

disableProximityInterrupt
 Pals2, [7](#)

disableProximityOffsetCompensation
 Pals2, [7](#)

enableAmbientLightInterrupt
 Pals2, [7](#)

enableColorCompensation
 Pals2, [8](#)

enablePeriodicMeasurements
 Pals2, [8](#)

enableProximityInterrupt
 Pals2, [8](#)

enableProximityOffsetCompensation
 Pals2, [9](#)

getIlluminance
 Pals2, [9](#)

getRawAmbientLight
 Pals2, [9](#)

getRawAmbientLightOnDemand
 Pals2, [9](#)

getRawProximity
 Pals2, [9](#)

getRawProximityOnDemand
 Pals2, [10](#)

Pals2, [5](#)

 begin, [7](#)

 disableAmbientLightInterrupt, [7](#)

 disableProximityInterrupt, [7](#)

 disableProximityOffsetCompensation, [7](#)

 enableAmbientLightInterrupt, [7](#)

 enableColorCompensation, [8](#)

 enablePeriodicMeasurements, [8](#)

 enableProximityInterrupt, [8](#)

 enableProximityOffsetCompensation, [9](#)

 getIlluminance, [9](#)

 getRawAmbientLight, [9](#)

 getRawAmbientLightOnDemand, [9](#)

 getRawProximity, [9](#)

 getRawProximityOnDemand, [10](#)

 setADCGain, [10](#)

 setAmbientLightMeasurementRate, [10](#)

 setInterruptPersistence, [10](#)

 setProximityMeasurementRate, [10](#)

 updateData, [11](#)

setADCGain
 Pals2, [10](#)

setAmbientLightMeasurementRate
 Pals2, [10](#)

setInterruptPersistence
 Pals2, [10](#)

setProximityMeasurementRate
 Pals2, [10](#)

updateData
 Pals2, [11](#)

Trademarks of Infineon Technologies AG

μHVIC™, μIPM™, μPFC™, AU-ConvertIR™, AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, CoolDP™, CoolGaN™, COOLiR™, CoolMOS™, CoolSET™, CoolSiC™, DAVE™, DI-POL™, DirectFET™, DrBlade™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, GaNpowIR™, HEXFET™, HITFET™, HybridPACK™, iMOTION™, IRAM™, ISOFACE™, IsoPACK™, LEDriviR™, LITIX™, MIPAQ™, ModSTACK™, my-d™, NovalithiC™, OPTIGA™, OptiMOS™, ORIGA™, PowIRaudio™, PowIRstage™, PrimePACK™, PrimeSTACK™, PROFET™, PRO-SiL™, RASIC™, REAL3™, SmartLEWIS™, SOLID FLASH™, SPOC™, StrongIRFET™, SuplIRBuck™, TEMPFET™, TRENCHSTOP™, TriCore™, UHVIC™, XHP™, XMC™.

Trademarks Update 2015-12-22

Other Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury