TAOS Inc.

is now

ams AG

The technical content of this TAOS document is still valid.

Contact information:

Headquarters:
ams AG
Tobelbader Strasse 30
8141 Premstaetten, Austria
Tel: +43 (0) 3136 500 0
e-Mail: ams_sales@ams.com

Please visit our website at **www.ams.com**

DESIGNER'S NOTEBOOK

Using Interrupts and the Persistence Filter

by Max Rosengarden
August 2012

Document Scope and Application

Products on the market today contain many different types of sensor ICs. As the use-cases of products become more refined and nuanced, sensor IC manufacturers should respond with refined feature sets. This document discusses two features embedded in ams digital light sensor ICs which enable product designers to handle more use-cases in fewer lines of code: interrupts and the persistence filter.

The *Interrupt* pin allows the light sensor to autonomously issue an interrupt signal when the sensor's measured light intensity is outside of a given range.

The *Persistence* filter allows the light sensor to reject high-frequency events (i.e. camera flashes, users momentarily obscuring the sensor's field of view), similar to a lowpass filter.

This document covers digital-output ALS, PROX, and RGB (color) sensors manufactured by ams AG.

Definitions

- ams AG – IC manufacturer; formerly known as Austriamicrosystems AG
- TAOS – Texas Advanced Optoelectronics Solutions, a former company, now part of ams AG
- ALS – Ambient Light Sensor
- PROX – Infrared Proximity Sensor
- HT – Abbreviation for *high threshold*
- LT – Abbreviation for *low threshold*
- 0b### – Binary notation where ### consists of 1's and 0's
- 0x### – Hexadecimal notation where ### consists of *0-9,A-F*

Objective and Implementation of Interrupt

Modern consumer electronics have set new standards for human-device interaction and device environmental awareness. Consumers now expect their products to respond to environmental changes, such as lighting or user-proximity changes in a quick and seamless fashion. The traditional way to check for changes is to repeatedly sample the output of sensors and compare the output against a threshold set. This is an inefficient use of time and energy. ams sensors have the ability to *internally* store thresholds, *internally* compare the output against thresholds, and assert an active low if an interrupt event occurs (thus waking up the rest of the system). The interrupt enables the part to be more autonomous, simplifies code, and saves power by eliminating the need to poll.

ams sensors have up to two independent internal interrupts: ALS (or RGB) and PROX. An interrupt is asserted if (and only if) sensor data is *outside* a set of thresholds ***x***-times-in-a-row (***x*** = set via I2C; see *persistence filter* section).

For ALS products, the interrupt and thresholds are tied to the output of *CH0*. For RGB products, the interrupt and thresholds are tied to the output of the *Clear* channel. For proximity sensing, the PROX interrupt can be tied to either CH0 or CH1 (see *PDIODE* bits in the *CONTROL* register).

The remainder of this document will discuss interrupts in the context of the TSL2772 ALS+PROX sensor.

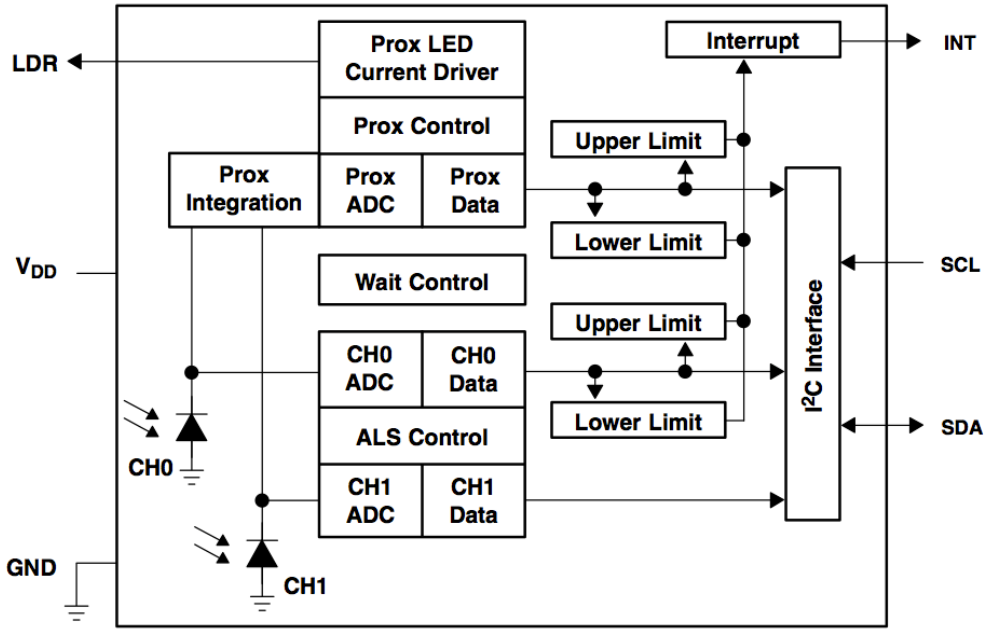

Figure 1 - Block diagram of TSL2772

In Figure 1, the reader will see a block diagram of the TSL2772 (notice the presence of both ALS and PROX thresholds).

Preparing For and Enabling Interrupts

There are two separate bits that enable interrupts: *AIEN* (ALS Interrupt Enable) and *PIEN* (PROX Interrupt Enable) located in the *Enable* register of the TSL2772. An interrupt will be asserted if the most recent data conversion is above the high threshold, or below the low threshold.

See a logic diagram below that describes the behavior of the interrupt mechanism.

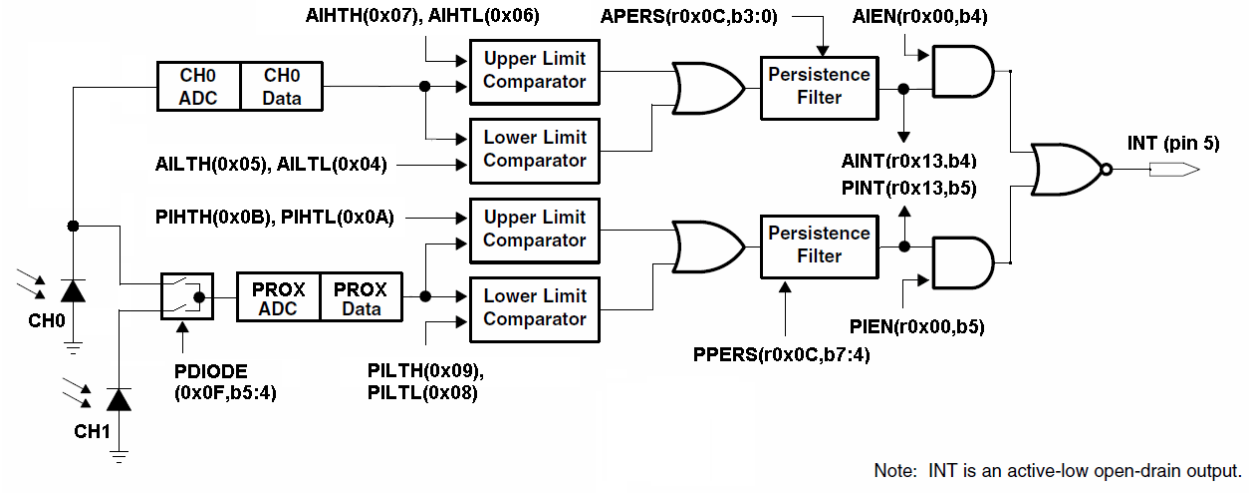

Figure 2 –TSL2772 logic diagram showing relationship between ALS, PROX, and INT pin

The persistence block will transmit a logic high signal if and only if the output of the and-gate (after the comparators) is high **x-in-a-row** times (up to 15), otherwise it outputs logic low. Before an interrupt can make its way to the *INT* pin, at least one of the bits *AIEN* (ALS interrupt enable) or *PIEN* (proximity interrupt enable) must be set. Both bits are located in the *Enable* register at location 0x00.

Clearing Interrupts

After servicing an interrupt and reading data, the user may then wish to clear the interrupt bits as to prepare for the next interrupt event. The *COMMAND* register is the gateway to clearing interrupts. The user clears the interrupts by first writing the slave address of the TSL2772, then a second byte 0b111001**xy** where **x**=1 to clear the ALS interrupt and **y**=1 to clear the PROX interrupt. Assuming the TSL2772's slave address is 0x39, the user would simply write 0x72[1] followed by 0xE7 to clear both interrupts.

Other Features Related to Interrupt

For products where it's critical to minimize power consumption, the sleep-after-interrupt feature may be of interest. The TSL2772 quickly and automatically transitions to "sleep mode" directly after detecting an interrupt. See the sleep-after-interrupt bit *SAI* in the *ENABLE* register.

The *PSAT* bit in the *STATUS* register indicates if the proximity subsystem has been saturated by an extreme amount of ambient IR during a proximity measurement. Clearing the proximity interrupt bit clears the *PSAT* bit.

Implementation of Interrupts across Product Families

Different devices in our family may behave slightly differently with respect to interrupts. While an interrupt *is* defined as an event outside the threshold window, some devices will not issue an interrupt while the output is *resting* on an upper or lower threshold.

The table below compares the behavior of the interrupt implementation across several light sensor products.

	ALS (or RGB) Sensing		Proximity Sensing		
	Issue interrupt if *CH0* (or *Clear)* data is…		Issue interrupt if PROX data is…		Minimum Cycle Time[2]
TSL2571	… less than LT	… greater than HT	n/a	n/a	5.4 ms
TSL2771	… less than LT	… greater than HT	… less than LT	…greater than HT	11 ms
TSL2572	… less than LT	… greater than HT	n/a	n/a	5.4 ms
TSL2772	… less than LT	… greater than HT	…less than LT	… greater than HT	13.8 ms
TMD2771	… less than LT	… greater than HT	…less than LT	… greater than HT	11 ms
TMD2772	… less than LT	… greater than HT	…less than LT	… greater than HT	13.8 ms
TSL258x	… less than LT	… greater than or equal to HT	n/a	n/a	2.7 ms
TCS347x	… less than LT	… greater than HT	n/a	n/a	5.4 ms

Table 1 - Description of interrupt implementation across different products

[1] 0x72 is the 7-bit slave address shifted into the top 7-bits of a byte, with bit-0 set as "0" to represent an I2C write. Leftshift(0x39,1)+0=(0x39*2)+0 = 0x72.

[2] "Minimum Cycle Time" is defined as the amount of time it takes to complete one set of ALS (or RGB) and PROX measurements before starting another set. The author assumes no wait states, 8 Prox pulses, 1 Prox integration cycle and 1 ALS integration cycle.

Objective and Implementation of Persistence Filter

In a real application, a sensor may experience a high-frequency "spike" which temporarily causes the sensor data to fall outside the thresholds and trigger an interrupt. An inadvertent hand movement can cause spikes in PROX readings, or driving under a small bridge may cause a dip in ALS readings. Both cases may result in an interrupt, but for some applications the user may wish to reject such high-frequency events and ensure interrupts don't occur.

Persistence filter bits give the user the option to require that up to 15-in-a-row measurements fall outside the threshold window before an interrupt can be generated. This functions as a pseudo-low pass filter.

Example—Using *Wait Time* in Conjunction with Persistence Filter Settings

If a user simply wishes to ensure that an interrupt is generated only after **x**-in-a-row measurements outside the threshold window, the *PERS* register allows the user to set **x**. If a user wishes to ensure an interrupt is generated after **y** seconds of continuously detecting an interrupt event, the user can use the "wait time" in conjunction with the persistence filter to ensure no short-term events trigger an interrupt.

For example, a designer may want to set up PROX and ALS to have the combined ability to issue interrupts in very different situations. A proximity interrupt event could be defined as *100ms* outside a threshold window, while an ALS event could be defined as *1s* outside a threshold window. Using the TSL2772's internal "wait time"[3] to insert a delay between measurements of approximately 87ms, the default ALS and PROX settings, ALS persistence filter value of 10 readings-in-a-row, the initial set of interrupt conditions are achieved. A proximity interrupt will be issued the instant a PROX conversion falls outside the threshold window, while it takes 10 ALS measurements in a row outside the threshold window. At 100ms per PROX (8ms[4])→ Internal wait[5] (87ms)→ALS (5ms[6])→*(repeat)* loop, 10 ALS conversions will take 1s.

[3] In the TSL2772 datasheet, see the WTIME register (0x03) and *Figure 14. Detailed State Diagram* to understand how to insert an automatic delay into the ALS/PROX measurement cycle
[4] Time required for one complete PROX conversion when using the TSL2772
[5] This programmable "wait" time is part of the chip's internal delay and is not a software delay
[6] Time required for one complete ALS conversion when using the TSL2772