TSL2591

Datasheet - Apr. 2013 - ams163.5

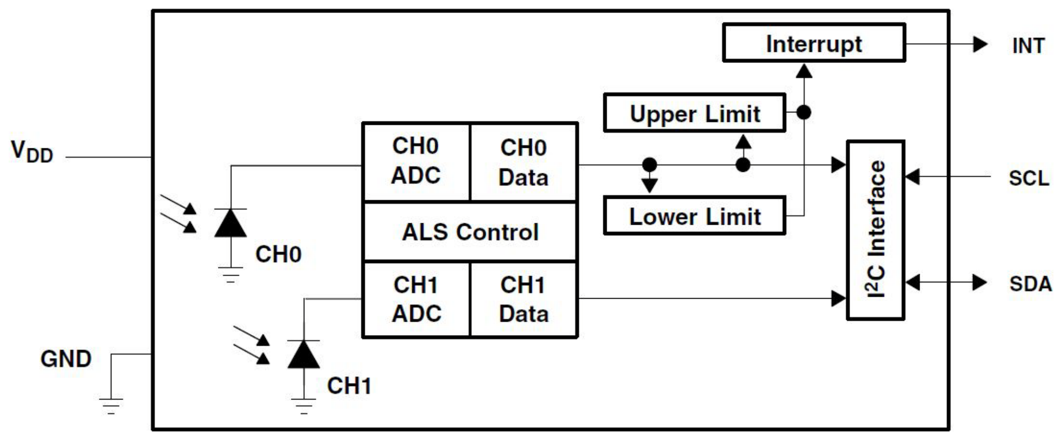
General Description

The TSL2591 is a very-high sensitivity light-to-digital converter that transforms light intensity into a digital signal output capable of direct I^2C interface. The device combines one broadband photodiode (visible plus infrared) and one infrared-responding photodiode on a single CMOS integrated circuit. Two integrating ADCs convert the photodiode currents into a digital output that represents the irradiance measured on each channel. This digital output can be input to a microprocessor where illuminance (ambient light level) in lux is derived using an empirical formula to approximate the human eye response. The TSL2591 supports a traditional level style interrupt that remains asserted until the firmware clears it.

Figure TSL2591 – 1:
Key Benefits and Features

Benefits	Features
Approximates Human Eye Response	Dual Diode
Flexible Operation	Programmable Analog Gain and Integration Time
Suited for Operation Behind Dark Glass	600M:1 Dynamic Range
Low Operating Overhead	• Two Internal Interrupt Sources • Programmable Upper and Lower Thresholds • One Interrupt Includes Programmable Persistence Filter
Low Power 3.0 µA Sleep State	User Selectable Sleep Mode
I^2C Fast Mode Compatible Interface	• Data Rates up to 400 kbit/s • Input Voltage Levels Compatible with 3.0V Bus

Figure TSL2591 – 2:
Block Diagram



Detailed Description

The TSL2591 contains two integrating analog-to-digital converters (ADC) that integrate currents from two photodiodes. Integration of both channels occurs simultaneously. Upon completion of the conversion cycle, the conversion result is transferred to the Channel 0 and Channel 1 data registers, respectively. The transfers are double-buffered to ensure that the integrity of the data is maintained. After the transfer, the device automatically begins the next integration cycle.

Communication with the device is accomplished through a standard, two-wire I^2C serial bus. Consequently, the TSL2591 can be easily connected to a microcontroller or embedded controller. No external circuitry is required for signal conditioning. Because the output of the device is digital, the output is effectively immune to noise when compared to an analog signal.

The TSL2591 also supports an interrupt feature that simplifies and improves system efficiency by eliminating the need to poll a sensor for a light intensity value. The primary purpose of the interrupt function is to detect a meaningful change in light intensity. The concept of a meaningful change can be defined by the user both in terms of light intensity and time, or persistence, of that change in intensity. The device has the ability to define two sets of thresholds, both above and below the current light level. An interrupt is generated when the value of a conversion exceeds either of these limits. One set of thresholds can be configured to trigger an interrupt only when the ambient light exceeds them for a configurable amount of time (persistence) while the other set can be configured to trigger an immediate interrupt.

Pin Assignment

The TSL2591 pin assignments are described below.

Figure TSL2591 – 3:
Pin Diagram

Package FN Dual Flat No-Lead (Top View): Package drawing is not to scale.

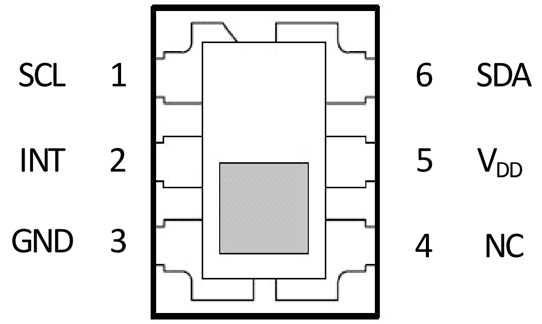


Figure TSL2591 – 4:
Pin Description

Pin Number	Pin Name	Description
1	SCL	I^2C serial clock input terminal
2	INT	Interrupt — open drain output (active low).
3	GND	Power supply ground. All voltages are referenced to GND.
4	NC	No connect — do not connect.
5	V_{DD}	Supply voltage
6	SDA	I^2C serial data I/O terminal

Ordering Information

Figure TSL2591 – 5:
Ordering Information

Ordering Code	Address	Interface	Delivery form
TSL25911FN	0x29	I^2C $V_{bus} = V_{DD}$ Interface	ODFN-6
TSL25913FN*	0x29	I^2C V_{bus} = 1.8V	ODFN-6

*Contact factory for availability.

Notes:

1. All products are RoHS compliant and ams green.
2. Buy our products or get free samples online at www.ams.com/ICdirect
3. Technical Support is available at www.ams.com/Technical-Support
4. For further information and requests, email us at sales@ams.com
5. (or) find your local distributor at www.ams.com/distributor
6. Please contact ams for alternate address device availability.

Absolute Maximum Ratings

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of the device at these or any other conditions beyond those indicated under "Operating Conditions" is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Figure TSL2591 – 6:
Absolute Maximum Ratings

Parameter	Min	Max	Units	Comments
Supply voltage, V_{DD}		3.8	V	All voltages are with respect to GND
Input terminal voltage	-0.5	3.8	V	
Output terminal voltage	-0.5	3.8	V	
Output terminal current	-1	20	mA	
Storage temperature range, T_{stg}	-40	85	ºC	
ESD tolerance, human body model		2000	V	

Electrical Characteristics

All limits are guaranteed. The parameters with min and max values are guaranteed with production tests or SQC (Statistical Quality Control) methods.

Figure TSL2591 – 7:
Recommended Operating Conditions

Symbol	Parameter	Min	Typ	Max	Units
V_{DD}	Supply voltage	2.7	3	3.6	V
T_A	Operating free-air temperature	-30		70	°C

Figure TSL2591 – 8:
Operating Characteristics, V_{DD}=3V, T_A=25°C (unless otherwise noted)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{DD}	Supply Current	Active Sleep state - no I^2C activity		275 2.3	325 4	µA
V_{OL}	INT, SDA output low voltage	3mA sink current 6mA sink current	0 0		0.4 0.6	V
I_{LEAK}	Leakage current, SDA, SCL, INT pins		-5		5	µA
V_{IH}	SCL, SDA input high voltage		0.7 V_{DD}			V
V_{IL}	SCL, SDA input low voltage				0.3 V_{DD}	V

Figure TSL2591 – 9:
ALS Characteristics, V_{DD}=3V, T_A=25°C, AGAIN = Max, AEN=1, (unless otherwise noted) (Notes 1, 2, 3),

Parameter	Conditions	Channel	Min	Typ	Max	Units
Dark ADC count value	$E_e = 0$, ATIME=000b (100ms)	CH0 CH1	0 0		25 25	counts
ADC integration time step size	ATIME = 000b (100ms)		95	101	108	ms
ADC number of integration steps (Note 4)			1		6	steps
ADC counts per step	ATIME = 000b (100ms)		0		37888	counts
ADC count value	ATIME = 101b (600ms)		0		65535	counts
ADC count value	White light (Note 2) $E_e = 4.98\ \mu W/cm^2$ ATIME = 000b (100 ms)	CH0 CH1	25500	30000 4996	34500	counts
	$\lambda_p = 850$ nm (Note 3) $E_e = 5.62\ \mu W/cm^2$, ATIME = 000b (100 ms)	CH0 CH1	25500	30000 19522	34500	counts
ADC count value ratio: CH1/CH0	White light (Note 2)		0.116	0.166	0.216	
	$\lambda_p = 850$ nm (Note 3)		0.456	0.652	0.848	
R_e Irradiance responsivity	White light (Note 2) ATIME = 000b (100 ms)	CH0 CH1		6024 1003		counts/ ($\mu W/cm^2$)
	$\lambda_p = 850$ nm (Note 3) ATIME = 000b (100 ms)	CH0 CH1		5338 3474		
Noise (Note 4)	White light (Note 2) $E_e = 4.98\ \mu W/cm^2$ ATIME = 000b (100 ms)	CH0		1	2	1 standard deviation
Gain scaling, relative to 1× gain setting	AGAIN = Low AGAIN = Med AGAIN = High AGAIN = Max			1 25 428 9876		×

Notes:

1. Optical measurements are made using small-angle incident radiation from light-emitting diode optical sources. Visible white LEDs and infrared 850 nm LEDs are used for final product testing for compatibility with high-volume production
2. The white LED irradiance is supplied by a white light-emitting diode with a nominal color temperature of 4000 K.
3. The 850 nm irradiance is supplied by a GaAs light-emitting diode with the following typical characteristics: peak wavelength $\lambda_p = 850$ nm and spectral halfwidth $\Delta\lambda½ = 42$ nm.
4. Parameter ensured by design and is not 100% tested.

Timing Characteristics

The timing characteristics of TSL2591 are given below.

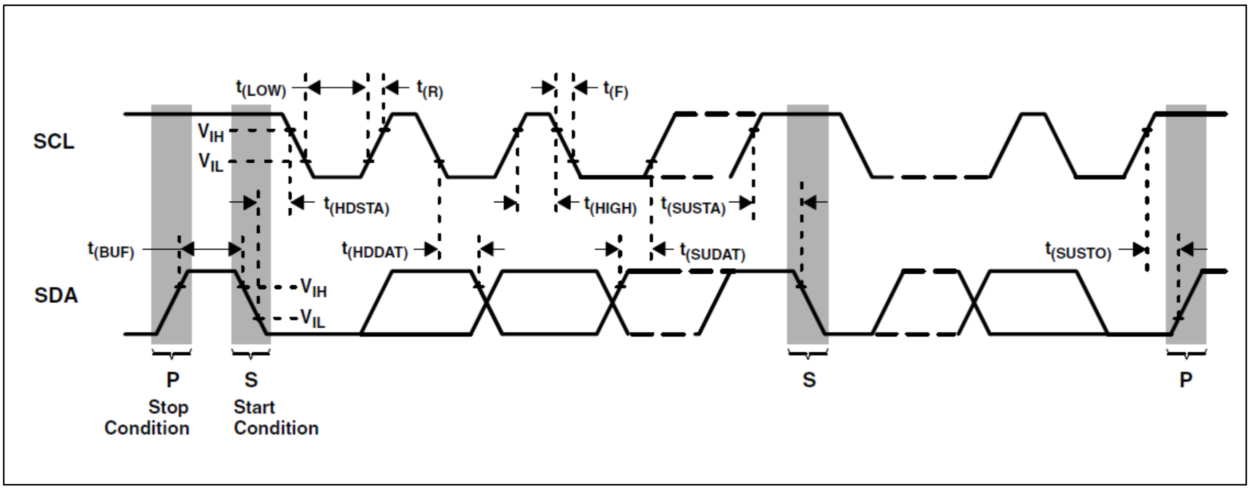
Figure TSL2591 – 10:
AC Electrical Characteristics, V_{DD} = 3 V, T_A = 25°C (unless otherwise noted)

Parameter†	Description	Min	Typ	Max	Units
$f_{(SCL)}$	Clock frequency (I^{2}C only)	0		400	kHz
$t_{(BUF)}$	Bus free time between start and stop condition	1.3			μs
$t_{(HDSTA)}$	Hold time after (repeated) start condition. After this period, the first clock is generated.	0.6			μs
$t_{(SUSTA)}$	Repeated start condition setup time	0.6			μs
$t_{(SUSTO)}$	Stop condition setup time	0.6			μs
$t_{(HDDAT)}$	Data hold time	0			μs
$t_{(SUDAT)}$	Data setup time	100			ns
$t_{(LOW)}$	SCL clock low period	1.3			μs
$t_{(HIGH)}$	SCL clock high period	0.6			μs
t_F	Clock/data fall time			300	ns
t_R	Clock/data rise time			300	ns
C_i	Input pin capacitance			10	pF

† Specified by design and characterization; not production tested.

Timing Diagrams

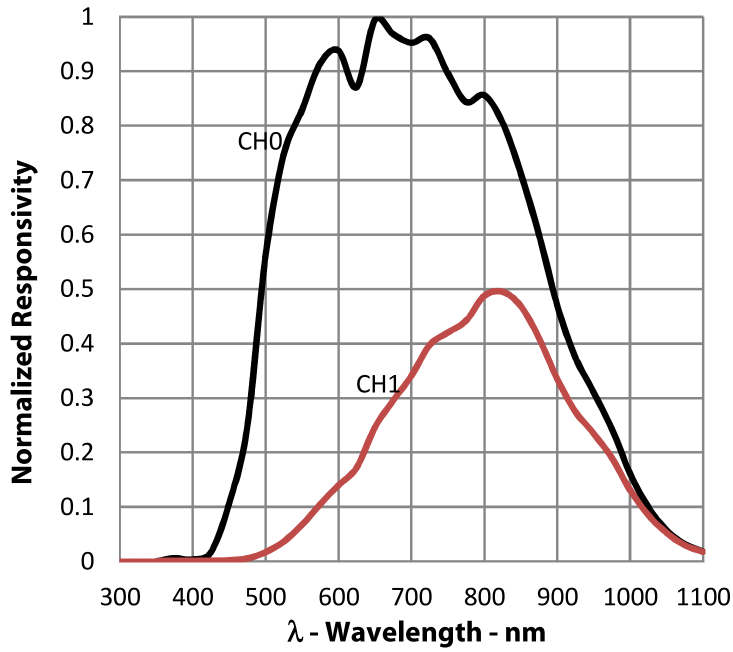
Figure TSL2591 – 11:
Parameter Measurement Information



Typical Operating Characteristics

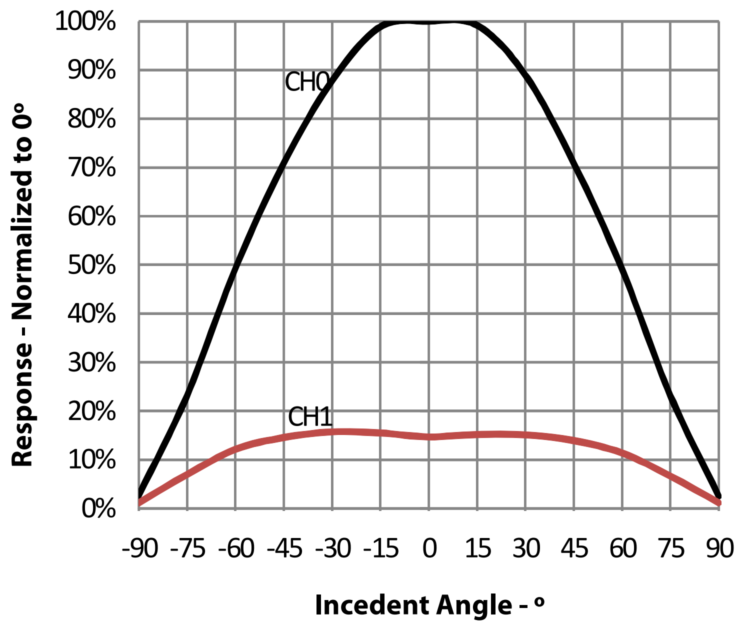
Spectral Responsivity: Two channel response allows for tunable illuminance (lux) calculation regardless of transmissivity of glass.

Figure TSL2591 – 12:
Spectral Responsivity



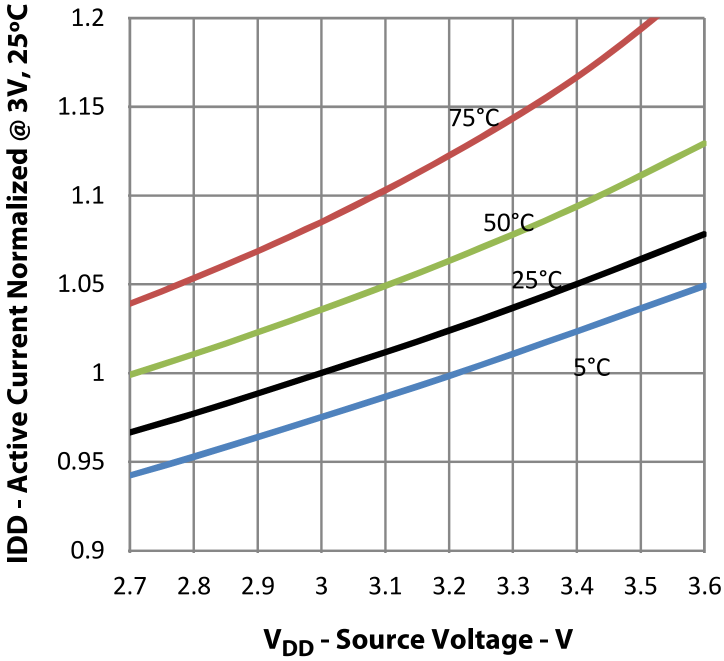
White LED Angular Response: Near cosine angular response for broadband white light sources.

Figure TSL2591 – 13:
White Normalized Responsivity vs. Angular Displacement



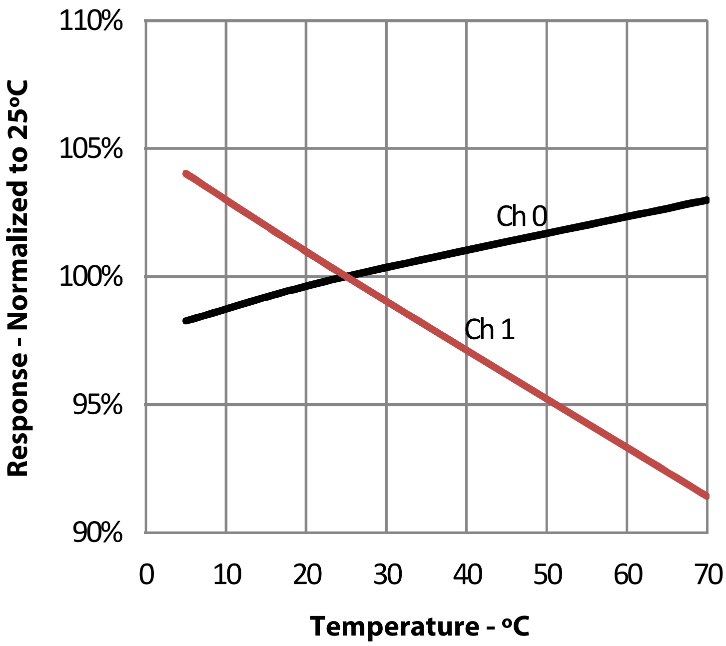
I_{DD} vs. V_{DD} vs. Temp: Effect of supply voltage and temperature on active current.

Figure TSL2591 – 14:
Normalized I_{DD} vs. V_{DD} and Temperature



White LED Response v Temp: Effect of temperature on the device response for a broadband white light source.

Figure TSL2591 – 15:
Response to White LED vs. Temperature



Register Description

The device is controlled and monitored by registers accessed through the I²C serial interface. These registers provide for a variety of control functions and can be read to determine results of the ADC conversions. The register set is summarized in Figure TSL2591 - 16.

Figure TSL2591 – 16:
Register Description

Address	Register Name	R/W	Register Function	Reset Value
--	COMMAND	W	Specifies Register Address	0x00
0x00	ENABLE	R/W	Enables states and interrupts	0x00
0x01	CONFIG	R/W	ALS gain and integration time configuration	0x00
0x04	AILTL	R/W	ALS interrupt low threshold low byte	0x00
0x05	AILTH	R/W	ALS interrupt low threshold high byte	0x00
0x06	AIHTL	R/W	ALS interrupt high threshold low byte	0x00
0x07	AIHTH	R/W	ALS interrupt high threshold high byte	0x00
0x08	NPAILTL	R/W	No Persist ALS interrupt low threshold low byte	0x00
0x09	NPAILTH	R/W	No Persist ALS interrupt low threshold high byte	0x00
0x0A	NPAIHTL	R/W	No Persist ALS interrupt high threshold low byte	0x00
0x0B	NPAIHTH	R/W	No Persist ALS interrupt high threshold high byte	0x00
0x0C	PERSIST	R/W	Interrupt persistence filter	0x00
0x11	PID	R	Package ID	--
0x12	ID	R	Device ID	ID
0x13	STATUS	R	Device status	0x00
0x14	C0DATAL	R	CH0 ADC low data byte	0x00
0x15	C0DATAH	R	CH0 ADC high data byte	0x00
0x16	C1DATAL	R	CH1 ADC low data byte	0x00
0x17	C1DATAH	R	CH1 ADC high data byte	0x00

Note: JGS-Stopped here.

Command Register

The COMMAND register specifies the address of the target register for future read and write operations, as well as issues special function commands.

7	6	5	4	3	2	1	0
CMD	TRANSACTION		ADDR/SF				

<table>
<tr><th>Fields</th><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>CMD</td><td>7</td><td colspan="2">Select Command Register. Must write as 1 when addressing COMMAND register.</td></tr>
<tr><td rowspan="6">TRANSACTION</td><td rowspan="6">6:5</td><td colspan="2">Select type of transaction to follow in subsequent data transfers</td></tr>
<tr><td>FIELD VALUE</td><td>DESCRIPTION</td></tr>
<tr><td>00</td><td>Reserved - Do not use</td></tr>
<tr><td>01</td><td>Normal Operation</td></tr>
<tr><td>10</td><td>Reserved – Do not use</td></tr>
<tr><td>11</td><td>Special Function – See description below</td></tr>
<tr><td rowspan="8">ADDR/SF</td><td rowspan="8">4:0</td><td colspan="2">Address field/special function field. Depending on the transaction type, see above, this field either specifies a special function command or selects the specific control-status-data register for subsequent read and write transactions. The field values listed below apply only to special function commands.</td></tr>
<tr><td>FIELD VALUE</td><td>DESCRIPTION</td></tr>
<tr><td>00100</td><td>Interrupt set – forces an interrupt</td></tr>
<tr><td>00110</td><td>Clears ALS interrupt</td></tr>
<tr><td>00111</td><td>Clears ALS and no persist ALS interrupt</td></tr>
<tr><td>01010</td><td>Clears no persist ALS interrupt</td></tr>
<tr><td>other</td><td>Reserved – Do not write</td></tr>
<tr><td colspan="2">The interrupt set special function command sets the interrupt bits in the status register (0x13). For the interrupt to be visible on the INT pin, one of the interrupt enable bits in the enable register (0x00) must be asserted.
The interrupt set special function must be cleared with an interrupt clear special function. The ALS interrupt clear special functions clear any pending interrupt(s) and are self-clearing.</td></tr>
</table>

Enable Register (0x00)

The ENABLE register is used to power the device on/off, enable functions and interrupts.

7	6	5	4	3	2	1	0
NPIEN	SAI	Reserved	AIEN	Reserved		AEN	PON

Fields	Bits	Description
NPIEN	7	No Persist Interrupt Enable. When asserted NP Threshold conditions will generate an interrupt, bypassing the persist filter.
SAI	6	Sleep after interrupt. When asserted, the device will power down at the end of an ALS cycle if an interrupt has been generated.
Reserved	5	Reserved. Write as 0.
AIEN	4	ALS Interrupt Enable. When asserted permits ALS interrupts to be generated, subject to the persist filter.
Reserved	3:2	Reserved. Write as 0.
AEN	1	ALS Enable. This field activates ALS function. Writing a one activates the ALS. Writing a zero disables the ALS.
PON	0	Power ON. This field activates the internal oscillator to permit the timers and ADC channels to operate. Writing a one activates the oscillator. Writing a zero disables the oscillator.

Control Register (0x01)

The CONTROL register is used to configure the ALS gain and integration time. In addition, a system reset is provided. Upon power up, the CONTROL register resets to 0x00.

7	6	5	4	3	2	1	0
SRESET	Reserved	AGAIN		Reserved	ATIME		

<table>
<tr><th>Fields</th><th>Bits</th><th colspan="3">Description</th></tr>
<tr><td>SRESET</td><td>7</td><td colspan="3">System reset. When asserted, the device will reset equivalent to a power-on reset. SRESET is self-clearing.</td></tr>
<tr><td>Reserved</td><td>6</td><td colspan="3">Reserved. Write as 0.</td></tr>
<tr><td rowspan="6">AGAIN</td><td rowspan="6">5:4</td><td colspan="3">ALS gain sets the gain of the internal integration amplifiers for both photodiode channels.</td></tr>
<tr><td>FIELD VALUE</td><td colspan="2">DESCRIPTION</td></tr>
<tr><td>00</td><td colspan="2">Low gain mode</td></tr>
<tr><td>01</td><td colspan="2">Medium gain mode</td></tr>
<tr><td>10</td><td colspan="2">High gain mode</td></tr>
<tr><td>11</td><td colspan="2">Maximum gain mode</td></tr>
<tr><td>Reserved</td><td>3</td><td colspan="3">Reserved. Write as 0.</td></tr>
<tr><td rowspan="8">ATIME</td><td rowspan="8">2:0</td><td colspan="3">ALS time sets the internal ADC integration time for both photodiode channels.</td></tr>
<tr><td>FIELD VALUE</td><td>INTEGRATION TIME</td><td>MAX COUNT</td></tr>
<tr><td>000</td><td>100 ms</td><td>37888</td></tr>
<tr><td>001</td><td>200 ms</td><td>65535</td></tr>
<tr><td>010</td><td>300 ms</td><td>65535</td></tr>
<tr><td>011</td><td>400 ms</td><td>65535</td></tr>
<tr><td>100</td><td>500 ms</td><td>65535</td></tr>
<tr><td>101</td><td>600 ms</td><td>65535</td></tr>
</table>

ALS Interrupt Threshold Register (0x04 – 0x0B)

The ALS interrupt threshold registers provide the values to be used as the high and low trigger points for the comparison function for interrupt generation. If C0DATA crosses below the low threshold specified, or above the higher threshold, an interrupt is asserted on the interrupt pin.

If the C0DATA exceeds the persist thresholds (registers: 0x04 – 0x07) for the number of persist cycles configured in the PERSIST register an interrupt will be triggered. If the C0DATA exceeds the no-persist thresholds (registers: 0x08 – 0x0B) an interrupt will be triggered immediately following the end of the current integration.

Note that while the interrupt is observable in the STATUS register (0x13), it is visible only on the INT pin when AIEN or NPIEN are enabled in the ENABLE register (0x00).

Upon power up, the interrupt threshold registers default to 0x00.

Register	Address	Bits	Description
AILTL	0x04	7:0	ALS low threshold lower byte
AILTH	0x05	7:0	ALS low threshold upper byte
AIHTL	0x06	7:0	ALS high threshold lower byte
AIHTH	0x07	7:0	ALS high threshold upper byte
NPAILTL	0x08	7:0	No Persist ALS low threshold lower byte
NPAILTH	0x09	7:0	No Persist ALS low threshold upper byte
NPAIHTL	0x0A	7:0	No Persist ALS high threshold lower byte
NPAIHTH	0x0B	7:0	No Persist ALS high threshold upper byte

PERSIST Register (0x0C)

The Interrupt persistence filter sets the number of consecutive out-of-range ALS cycles necessary to generate an interrupt. Out-of-range is determined by comparing C0DATA (0x14 and 0x15) to the interrupt threshold registers (0x04 - 0x07). Note that the no-persist ALS interrupt is not affected by the interrupt persistence filter. Upon power up, the interrupt persistence filter register resets to 0x00.

7	6	5	4	3	2	1	0
Reserved				APERS			

Field	Bits	Description	
Reserved	7:4	Reserved. Write as 0.	
APERS	3:0	ALS interrupt persistence filter	
		FIELD VALUE	**PERSISTENCE**
		0000	Every ALS cycle generates an interrupt
		0001	Any value outside of threshold range
		0010	2 consecutive values out of range
		0011	3 consecutive values out of range
		0100	5 consecutive values out of range
		0101	10 consecutive values out of range
		0110	15 consecutive values out of range
		0111	20 consecutive values out of range
		1000	25 consecutive values out of range
		1001	30 consecutive values out of range
		1010	35 consecutive values out of range
		1011	40 consecutive values out of range
		1100	45 consecutive values out of range
		1101	50 consecutive values out of range
		1110	55 consecutive values out of range
		1111	60 consecutive values out of range

PID Register (0x11)

The PID register provides an identification of the devices package. This register is a read-only register whose value never changes.

7	6	5	4	3	2	1	0
Reserved		PACKAGEID		Reserved			

Field	Bits	Description
Reserved	7:6	Reserved.
PID	5:4	Package Identification = 00
Reserved	3:0	Reserved.

ID Register (I0x12)

The ID register provides the device identification. This register is a read-only register whose value never changes.

7	6	5	4	3	2	1	0
ID							

Field	Bits	Description
ID	7:0	Device Identification = 0x50

Status Register (0x13)

The Status Register provides the internal status of the device. This register is read only.

7	6	5	4	3	2	1	0
Reserved		NPINTR	AINT	Reserved			AVALID

Field	Bits	Description
Reserved	7:6	Reserved. Write at zero.
NPINTR	5	No-persist Interrupt. Indicates that the device has encountered a no-persist interrupt condition.
AINT	4	ALS Interrupt. Indicates that the device is asserting an ALS interrupt.
Reserved	3:1	Reserved.
AVALID	0	ALS Valid. Indicates that the ADC channels have completed an integration cycle since the AEN bit was asserted.

ALS Data Register (0x14 - 0x17)

ALS data is stored as two 16-bit values; one for each channel. When the lower byte of either channel is read, the upper byte of the same channel is latched into a shadow register. The shadow register ensures that both bytes are the result of the same ALS integration cycle, even if additional integration cycles occur between the lower byte and upper byte register readings.

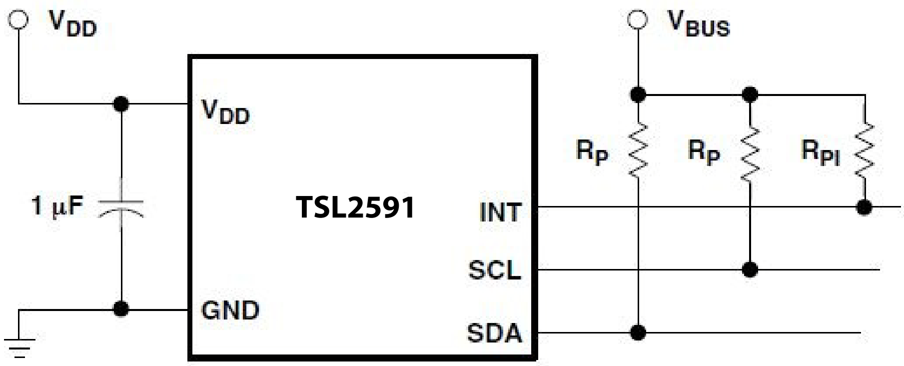
Each channel independently operates the upper byte shadow register. So to minimize the potential for skew between CH0 and CH1 data, it is recommended to read all four ADC bytes in sequence. The simplest way to accomplish this is to perform a four-byte I^2C read operation using the auto-increment protocol, which is set in the Command register TRANSACTION field.

Register	Address	Bits	Description
C0DATAL	0x14	7:0	ALS CH0 data low byte
C0DATAH	0x15	7:0	ALS CH0 data high byte
C1DATAL	0x16	7:0	ALS CH1 data low byte
C1DATAH	0x17	7:0	ALS CH1 data high byte

Application Information

Figure TSL2591 - 17 shows a typical hardware application circuit. A 1-µF low-ESR decoupling capacitor should be placed as close as possible to the V_{DD} pin. V_{BUS} in this figure refers to the I^2C bus voltage, which is equal to V_{DD}.

Figure TSL2591 – 17:
Typical Application Hardware Circuit

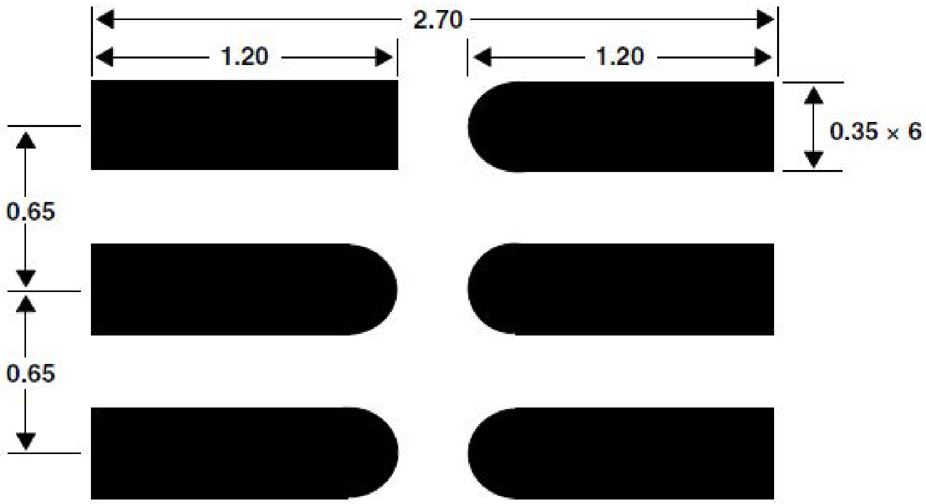


The I^2C signals and the Interrupt are open-drain outputs and require pull-up resistors. The pull-up resistor (RP) value is a function of the I^2C bus speed, the I^2C bus voltage, and the capacitive load. The ams EVM running at 400 kbps, uses 1.5-kΩ resistors. A 10-kΩ pull-up resistor (RPI) can be used for the interrupt line.

PCB Pad Layout

Suggested land pattern based on the IPC−7351B Generic Requirements for Surface Mount Design and Land Pattern Standard (2010) for the small outline no-lead (SON) package is shown in Figure TSL2591 - 18.

Figure TSL2591 – 18:
Suggested FN Package PCB Layout (Top View)

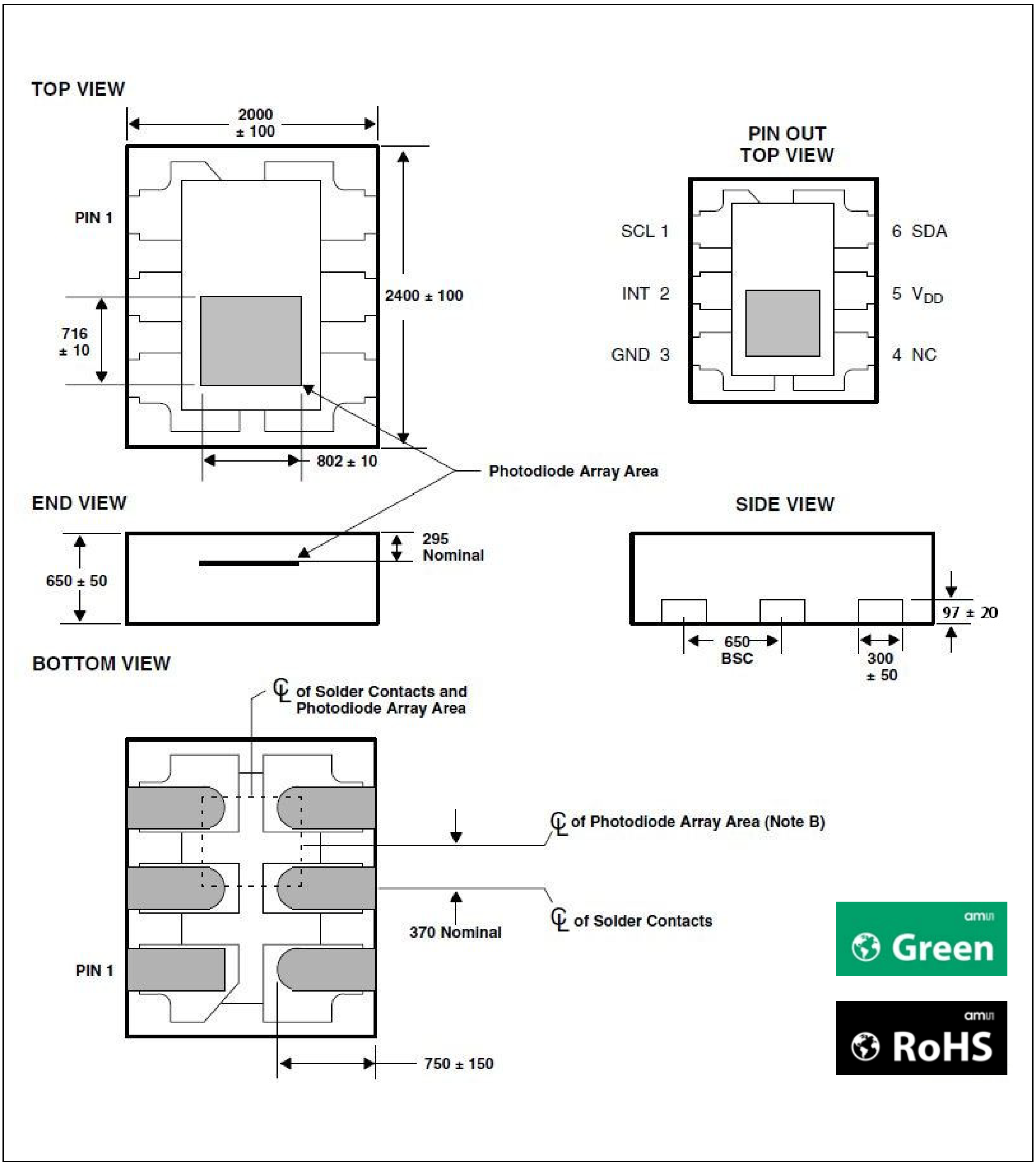


Notes:

1. All linear dimensions are in millimeters.
2. This drawing is subject to change without notice.

Package Drawings and Markings

Figure TSL2591 – 19:
FN Package – Dual Flat No-Lead Packaging Configuration

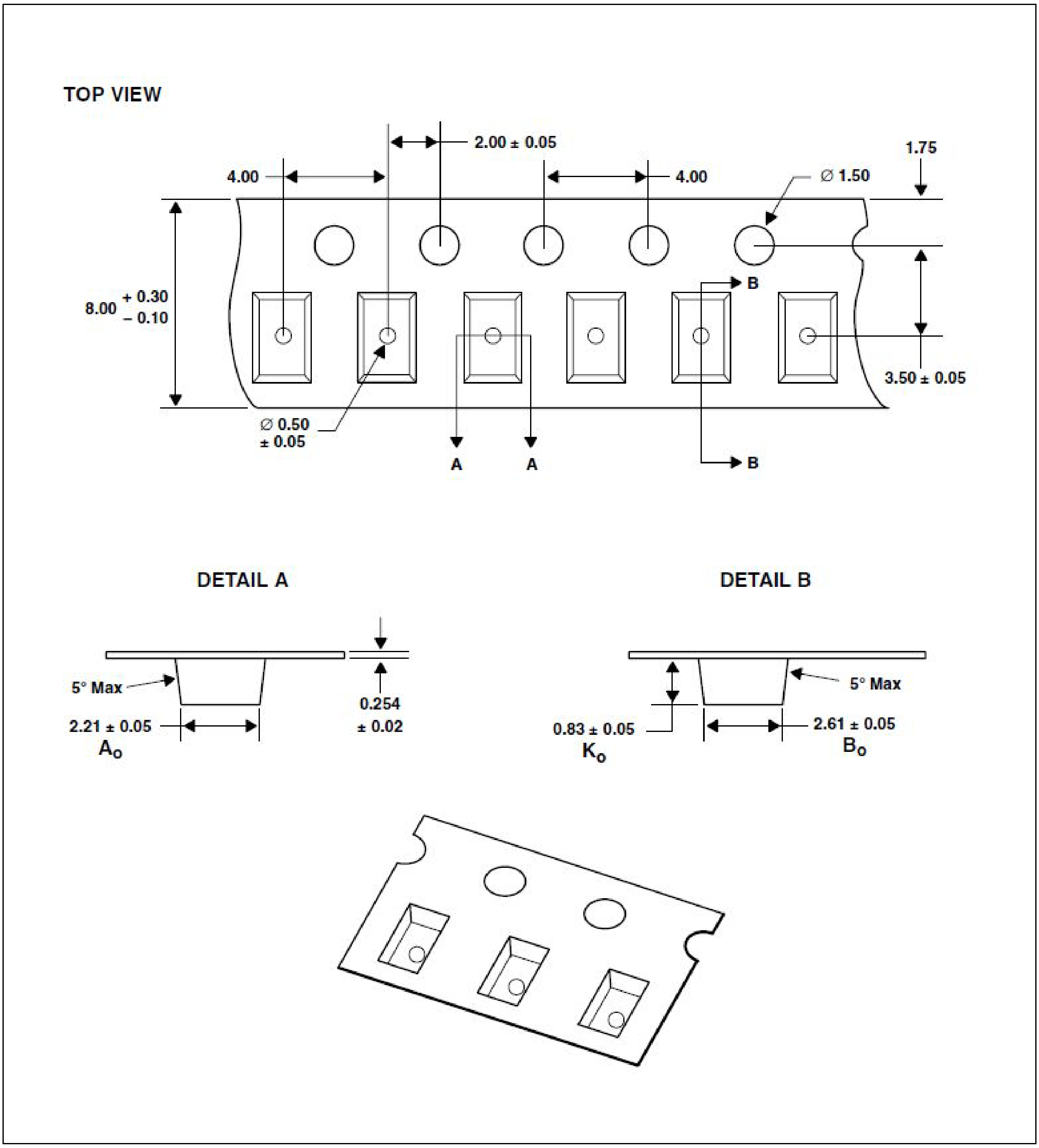


Notes:

1. All linear dimensions are in micrometers.
2. The die is centered within the package within a tolerence of ±75 µm.
3. Package top surface is molded with an electrically non-conductive clear plastic compound having an index of refraction of 1.55.
4. Contact finish is copper alloy A194 with pre-plated NIPdAu lead finish.
5. This package contains no lead (Pb).
6. This drawing is subject to change without notice.

Mechanical Data

Figure TSL2591 – 20:
FN Package Carrier Tape and Reel Information



Notes:

1. All linear dimensions are in millimeters. Dimension tolerance is ± 0.10 mm unless otherwise noted.
2. The dimensions on this drawing are for illustrative purposes only. Dimensions of an actual carrier may vary slightly.
3. Symbols on drawing A_O, B_O and K_O are defined in ANSI EIA Standard 481-B 2001.
4. Each reel is 178 millimeters in diameter and contains 3500 parts.
5. ams packaging tape and reel conform to the requirements of EIA Standard 481 - B.
6. In accordance with EIA Standard, device pin 1 is located next to the sprocket holes in the tape.
7. This drawing is subject to change without notice.

Soldering Information

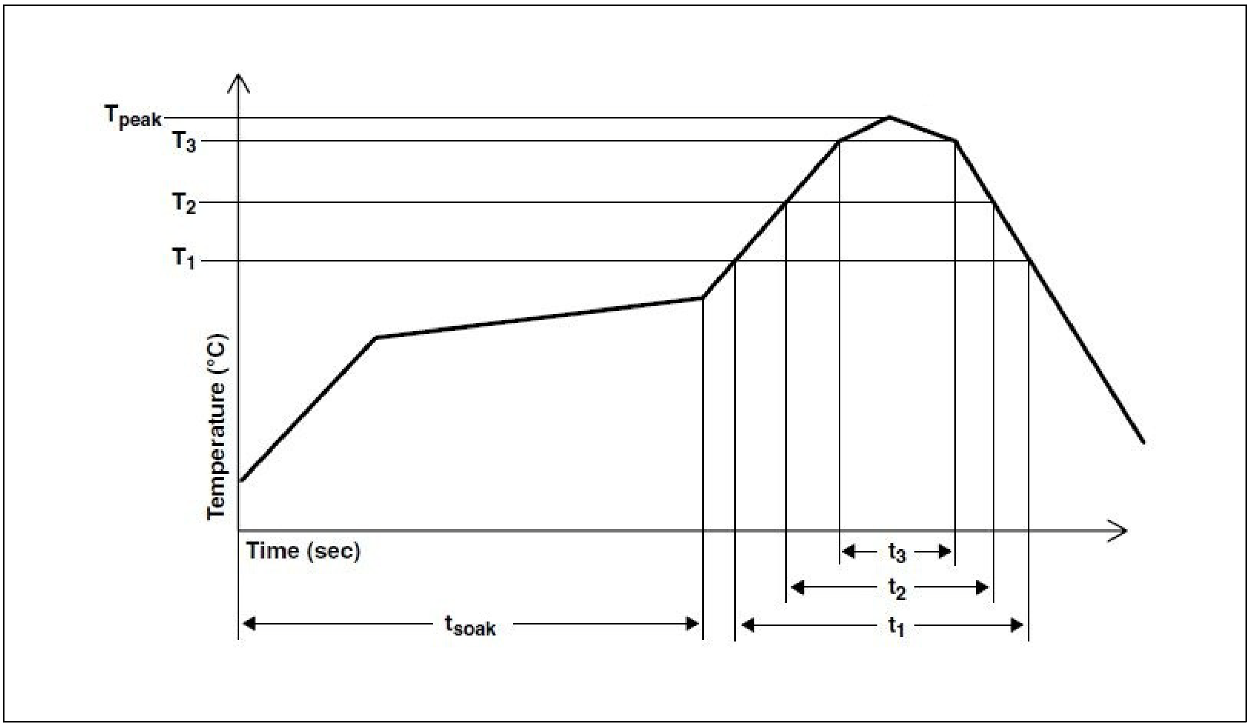
The package has been tested and has demonstrated an ability to be reflow soldered to a PCB substrate.

The solder reflow profile describes the expected maximum heat exposure of components during the solder reflow process of product on a PCB. Temperature is measured on top of component. The components should be limited to a maximum of three passes through this solder reflow profile.

Figure TSL2591 – 21: Solder Reflow Profile

Parameter	Reference	Device
Average temperature gradient in preheating		2.5 °C/sec
Soak time	t_{soak}	2 to 3 minutes
Time above 217 °C (T1)	t_1	Max 60 sec
Time above 230 °C (T2)	t_2	Max 50 sec
Time above T_{peak} - 10 °C (T3)	t_3	Max 10 sec
Peak temperature in reflow	T_{peak}	260 °C
Temperature gradient in cooling		Max -5 °C/sec

Figure TSL2591 – 22: Solder Reflow Profile Graph



Note: Not to scale – for reference only.

Storage Information

Moisture Sensitivity

Optical characteristics of the device can be adversely affected during the soldering process by the release and vaporization of moisture that has been previously absorbed into the package. To ensure the package contains the smallest amount of absorbed moisture possible, each device is baked prior to being dry packed for shipping.

Devices are dry packed in a sealed aluminized envelope called a moisture-barrier bag with silica gel to protect them from ambient moisture during shipping, handling, and storage before use.

Shelf Life

The calculated shelf life of the device in an unopened moisture barrier bag is 12 months from the date code on the bag when stored under the following conditions:

- Shelf Life: 12 months
- Ambient Temperature: < 40°C
- Relative Humidity: < 90%

Rebaking of the devices will be required if the devices exceed the 12 month shelf life or the Humidity Indicator Card shows that the devices were exposed to conditions beyond the allowable moisture region.

Floor Life

The FN package has been assigned a moisture sensitivity level of MSL 3. As a result, the floor life of devices removed from the moisture barrier bag is 168 hours from the time the bag was opened, provided that the devices are stored under the following conditions:

- Floor Life: 168 hours
- Ambient Temperature: < 30°C
- Relative Humidity: < 60%

If the floor life or the temperature/humidity conditions have been exceeded, the devices must be rebaked prior to solder reflow or dry packing.

Rebaking Instructions

When the shelf life or floor life limits have been exceeded, rebake at 50°C for 12 hours.

RoHS Compliant and ams Green Statement

The term RoHS complaint means that ams products fully comply with current RoHS directive. Our semiconductor products do not contain any chemicals for all 6 substance categories, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, RoHS compliant products are suitable for use in specified lead-free processes. ams Green means RoHS compliant and no Sb/Br). ams defines Green that additionally to RoHS compliance our products are free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material).

Important Information and Disclaimer The information provided in this statement represents ams knowledge and belief as of the date that it is provided. ams bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. ams has taken and continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. ams and ams suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

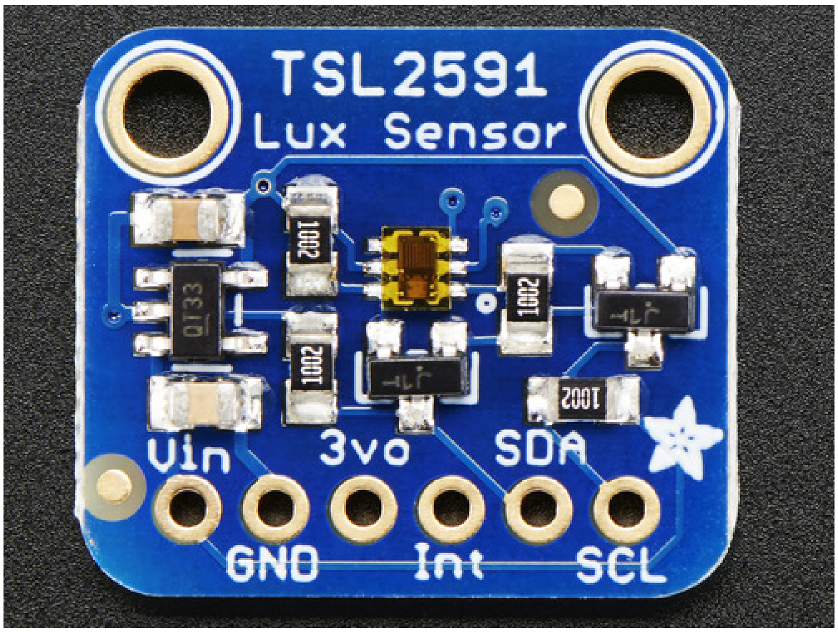
Copyrights

Disclaimer

adafruit learning system

Adafruit TSL2591 High Dynamic Range Digital Light Sensor

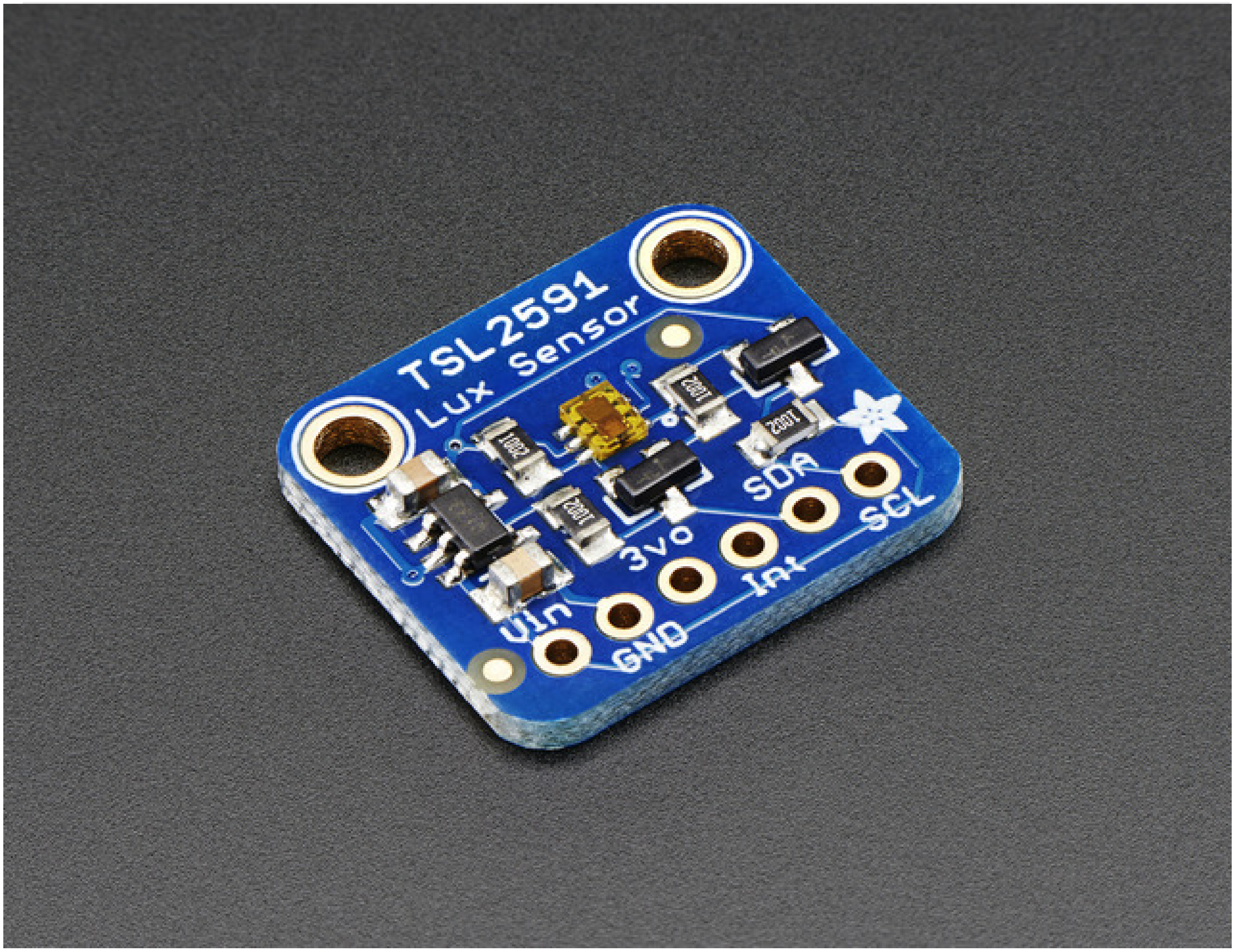
Created by lady ada



Last updated on 2014-07-11 02:15:07 PM EDT

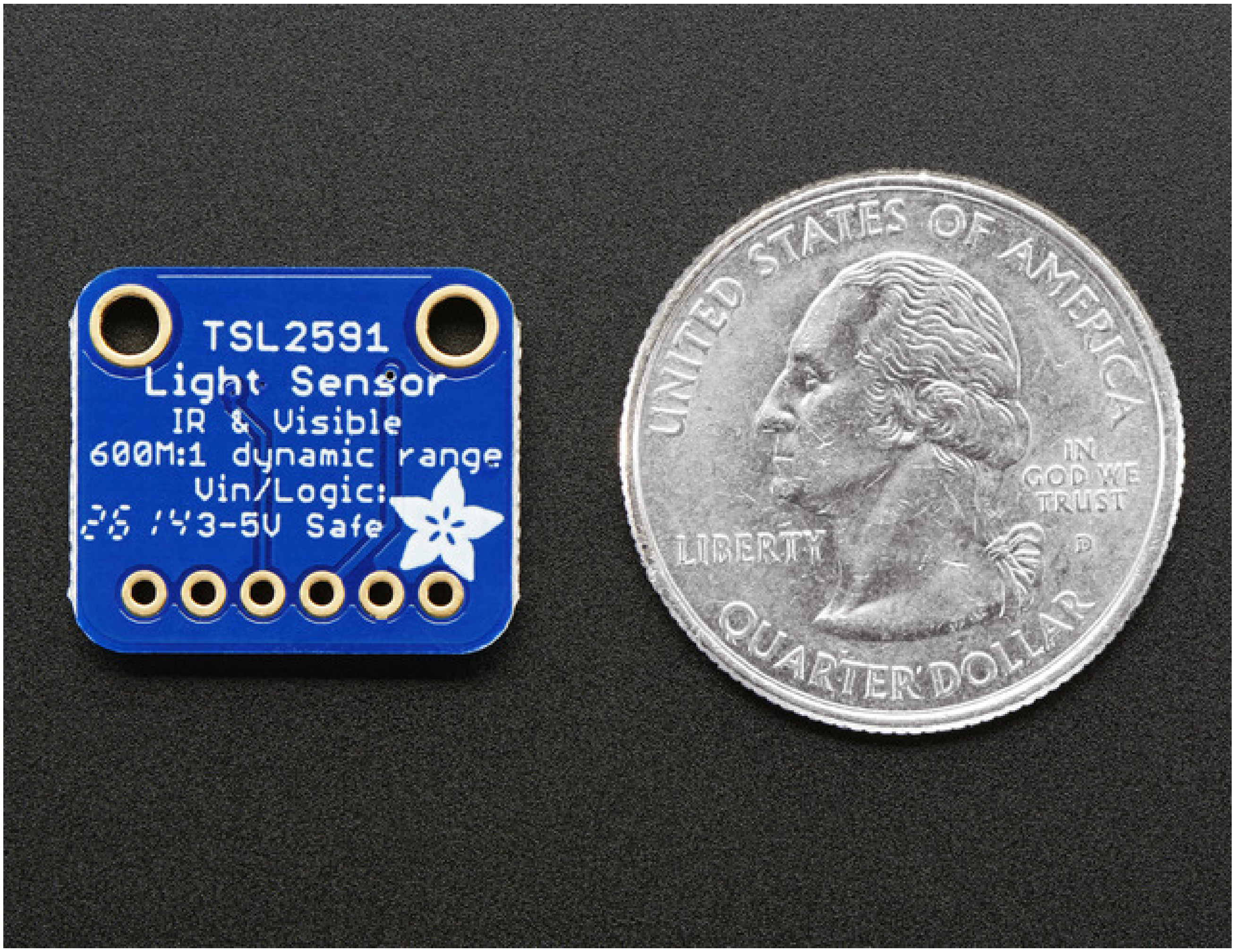
Guide Contents

Overview

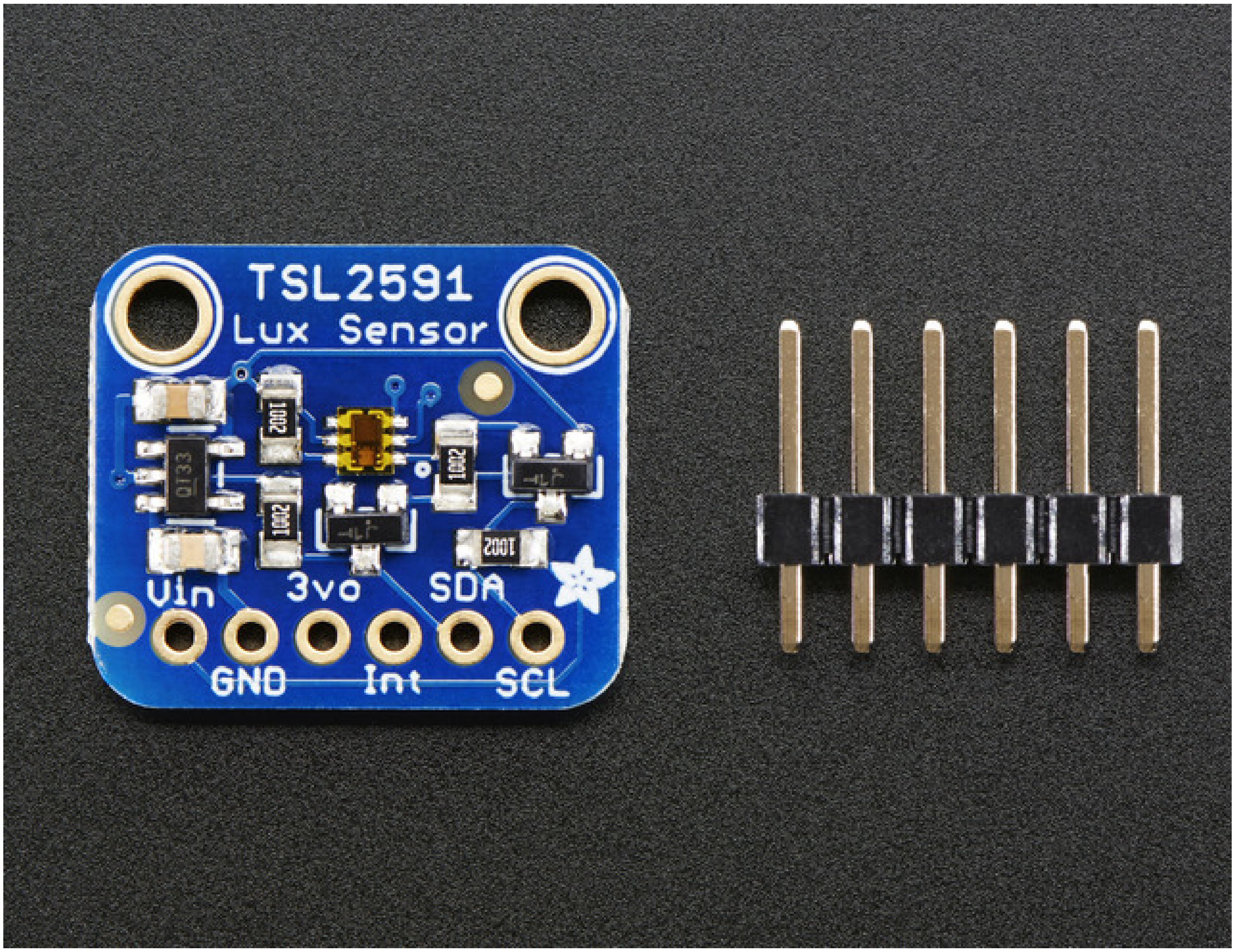


When the future is dazzlingly-bright, this ultra-high-range luminosity sensor will help you measure it. The TSL2591 luminosity sensor is an advanced digital light sensor, ideal for use in a wide range of light situations. Compared to low cost CdS cells, this sensor is more precise, allowing for exact lux calculations and can be configured for different gain/timing ranges to detect light ranges from up to 188uLux up to 88,000 Lux on the fly.

The best part of this sensor is that it **contains both infrared and full spectrum diodes**! That means you can separately measure infrared, full-spectrum or human-visible light. Most sensors can only detect one or the other, which does not accurately represent what human eyes see (since we cannot perceive the IR light that is detected by most photo diodes)



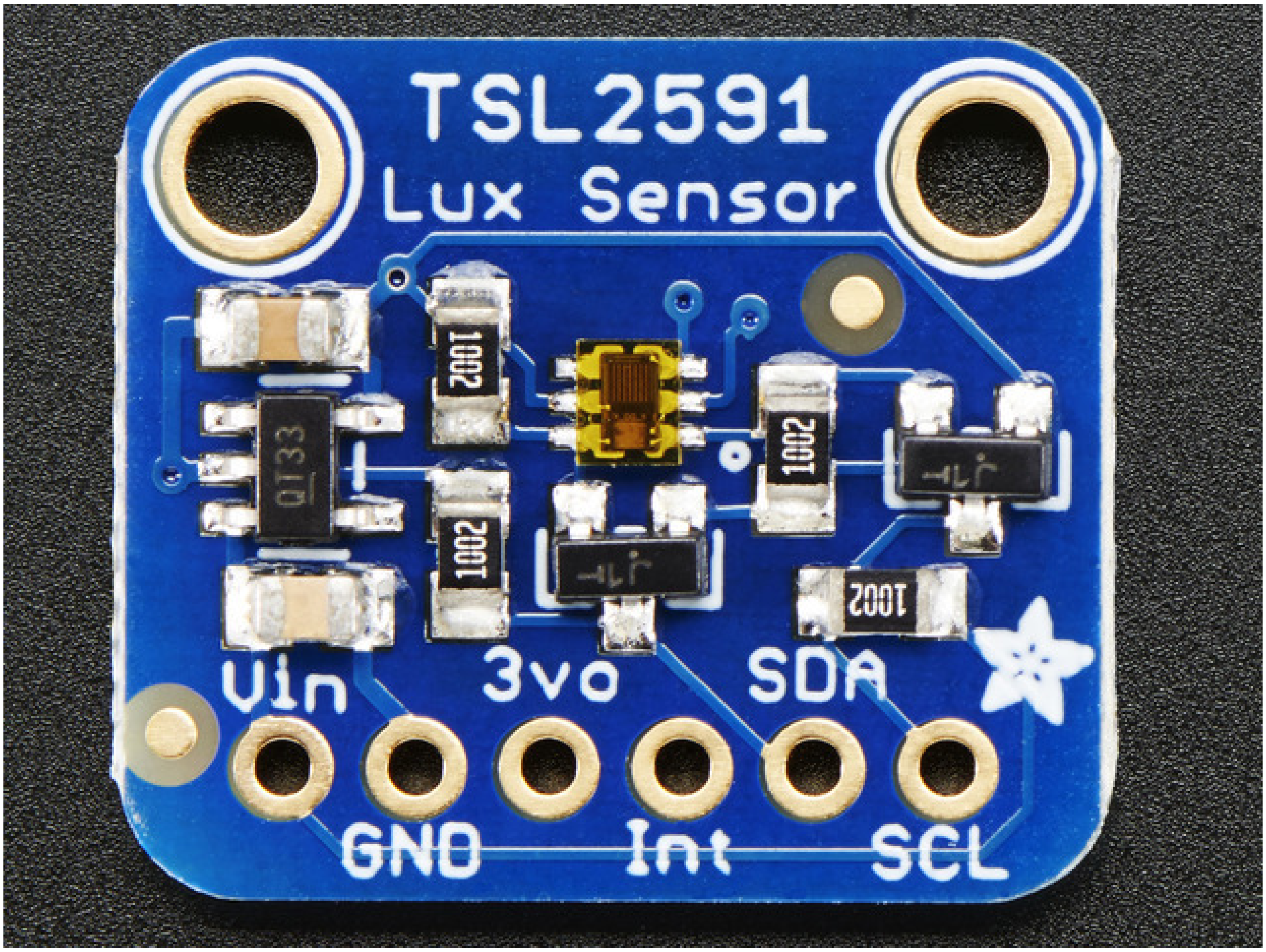
This sensor is much like the TSL2561 but with a wider range (and the interface code is different). This sensor has a massive 600,000,000:1 dynamic range! Unlike the TSL2561 you cannot change the I2C address either, so keep that in mind.



The built in ADC means you can use this with any microcontroller, even if it doesn't have analog inputs. The current draw is extremely low, so its great for low power data-logging systems. about 0.4mA when actively sensing, and less than 5 uA when in power-down mode.

Pinouts

The TSL2591 is a I2C sensor. That means it uses the two I2C data/clock wires available on most microcontrollers, and can share those pins with other sensors as long as they don't have an address collision. For future reference, the I2C address is **0x29** and you *can't* change it!



Power Pins:

- **Vin** - this is the power pin. Since the chip uses 3 VDC, we have included a voltage regulator on board that will take 3-5VDC and safely convert it down. To power the board, give it the same power as the logic level of your microcontroller - e.g. for a 5V micro like Arduino, use 5V
- **3vo** - this is the 3.3V output from the voltage regulator, you can grab up to 100mA from this if you like
- **GND** - common ground for power and logic

(http://adafru.it/dGy)I2C Logic pins:

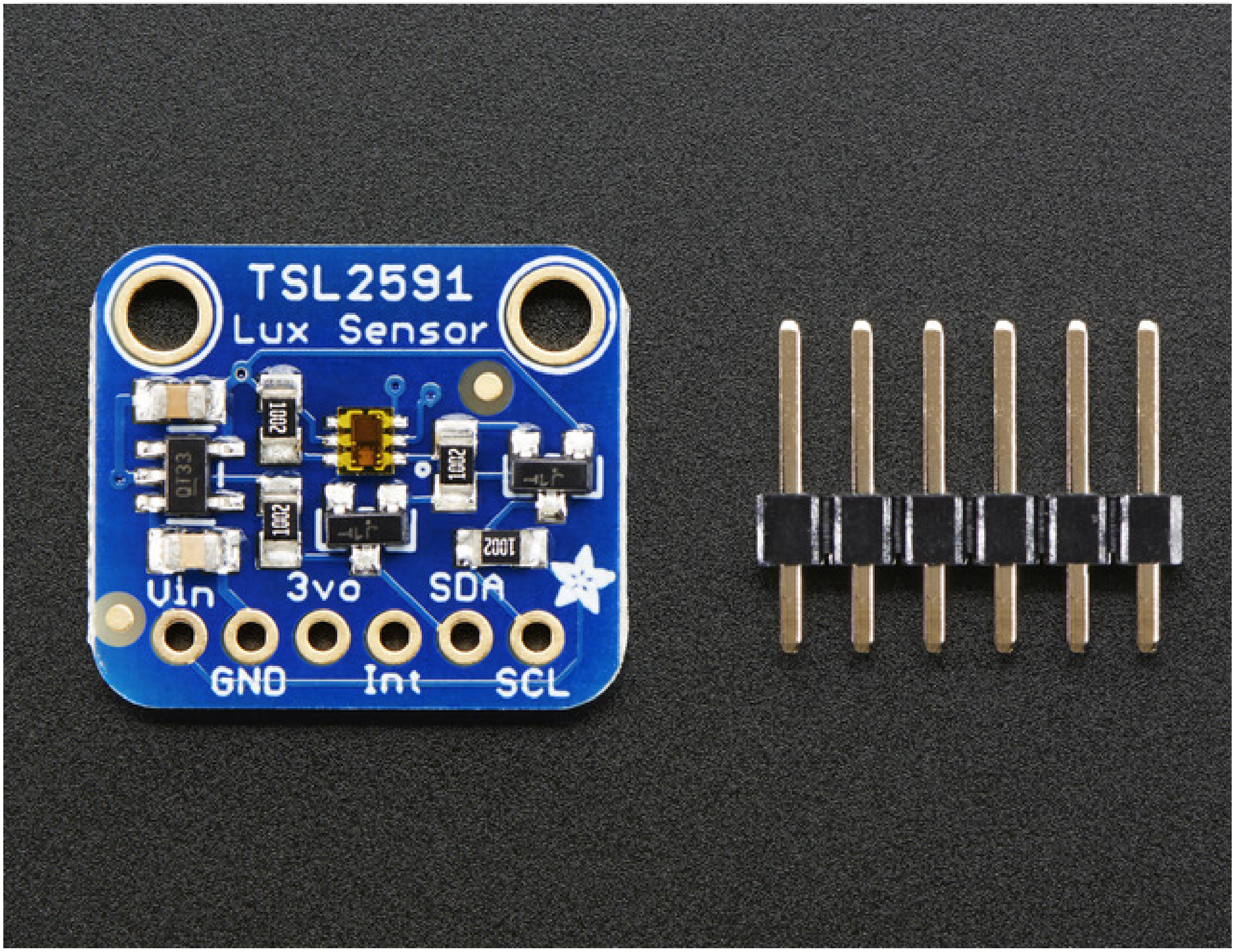
- **SCL** - I2C clock pin, connect to your microcontrollers I2C clock line.

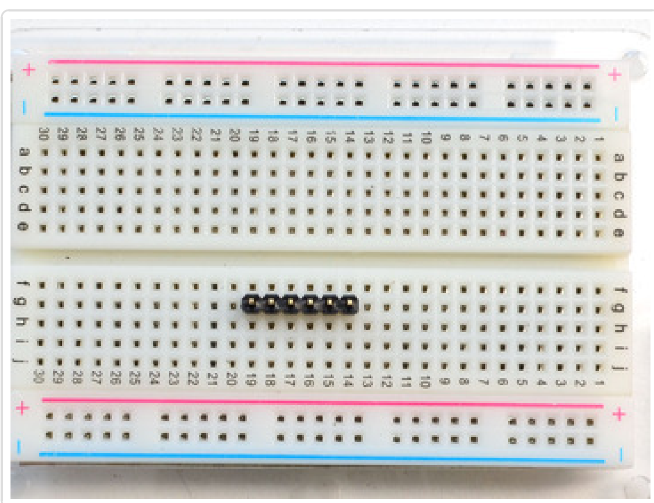
- **SDA** - I2C data pin, connect to your microcontrollers I2C data line.

Other Pins:

- **INT** - this is the INTerrupt pin from the sensor. It can be programmed to do a couple different things by noodling with the i2c registers. For example trigger when a conversion is done, or when the light level has changed a lot, etc. We don't have library support for this pin

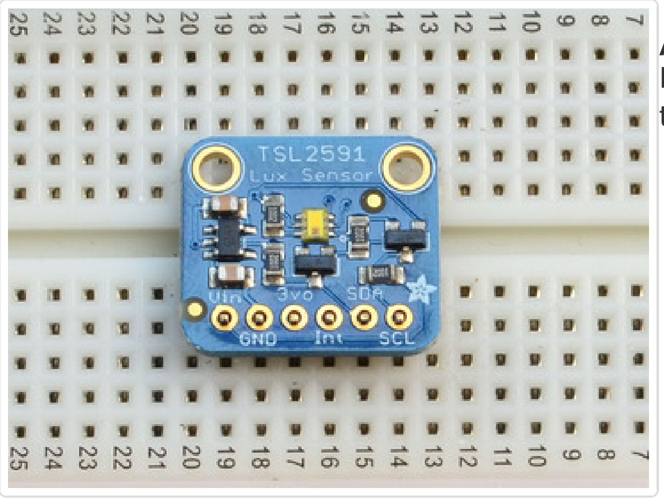
Assembly





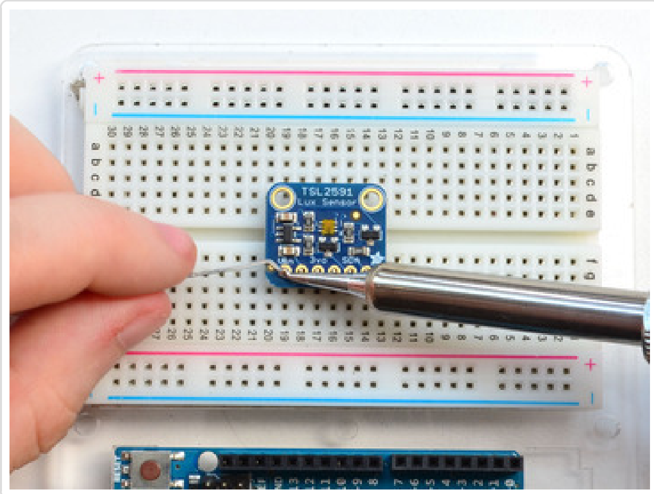
Prepare the header strip:

Cut the strip to length if necessary. It will be easier to solder if you insert it into a breadboard - **long pins down**



Add the breakout board:

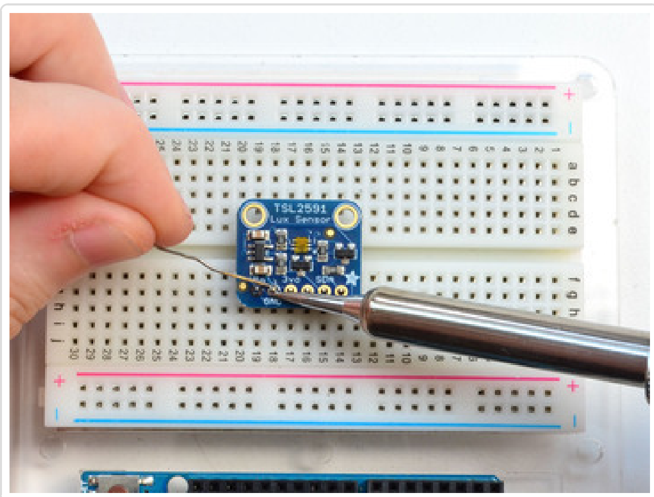
Place the breakout board over the pins so that the short pins poke through the breakout pads

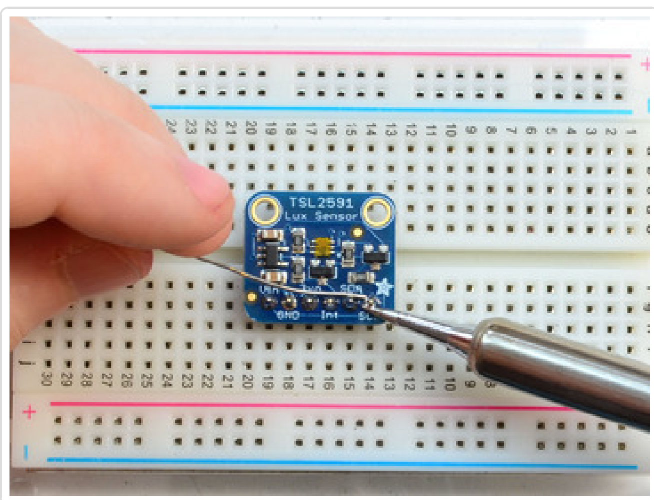


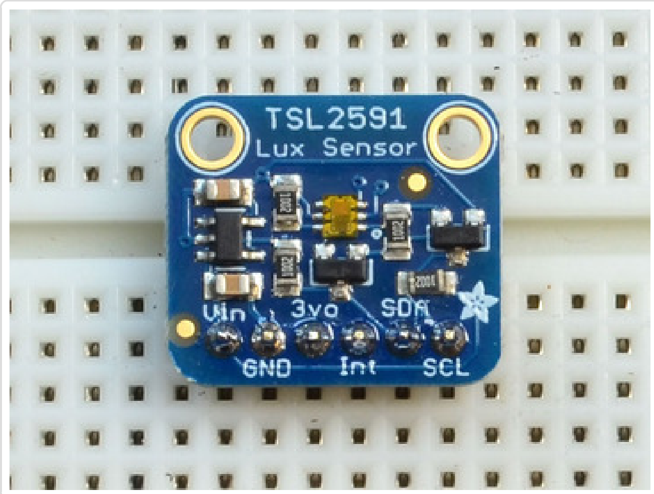
And Solder!

Be sure to solder all pins for reliable electrical contact.

(For tips on soldering, be sure to check out our Guide to Excellent Soldering (http://adafru.it/aTk)).



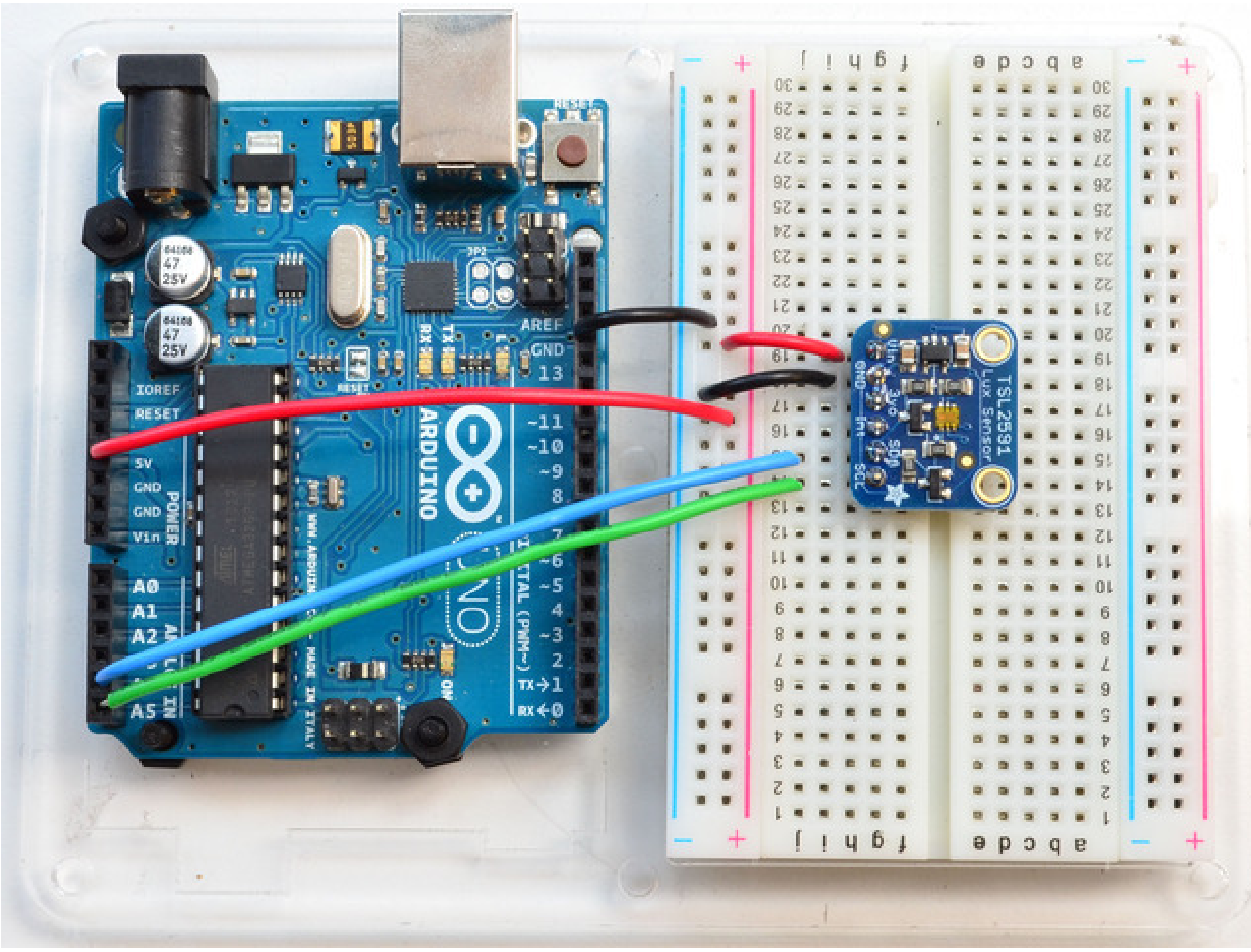




You're done! Check your solder joints visually and continue onto the next steps

Wiring & Test

You can easily wire this breakout to any microcontroller, we'll be using an Arduino. For another kind of microcontroller, just make sure it has I2C, then port the code - its pretty simple stuff!



(http://adafru.it/dBn)

- Connect **Vin** to the power supply, 3-5V is fine. Use the same voltage that the microcontroller logic is based off of. For most Arduinos, that is 5V
- Connect **GND** to common power/data ground
- Connect the **SCL** pin to the I2C clock **SCL** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A5**, on a Mega it is also known as **digital 21** and on a Leonardo/Micro, **digital 3**
- Connect the **SDA** pin to the I2C data **SDA** pin on your Arduino. On an UNO & '328 based Arduino, this is also known as **A4**, on a Mega it is also known as **digital 20** and on a Leonardo/Micro, **digital 2**

The TSL2591 has a default I2C address of **0x29** and cannot be changed!

Download Adafruit_TSL2591

To begin reading sensor data, you will need to download Adafruit_TSL2591_Library from our github repository (http://adafru.it/dGz). You can do that by visiting the github repo and manually downloading or, easier, just click this button to download the zip

Download Adafruit TSL2591 Library

http://adafru.it/dGA

Rename the uncompressed folder **Adafruit_TSL2591** and check that the **Adafruit_TSL2591** folder contains **Adafruit_TSL2591.cpp** and **Adafruit_TSL2591.h**

Place the **Adafruit_TSL2591** library folder your **arduinosketchfolder/libraries/** folder. You may need to create the **libraries** subfolder if its your first library. Restart the IDE.

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (http://adafru.it/aYM)

Download Adafruit_Sensor

The TSL2591 library uses the Adafruit_Sensor support backend so that readings can be normalized between sensors. You can grab Adafruit_Sensor from the github repo (http://adafru.it/aZm) or just click the button below.

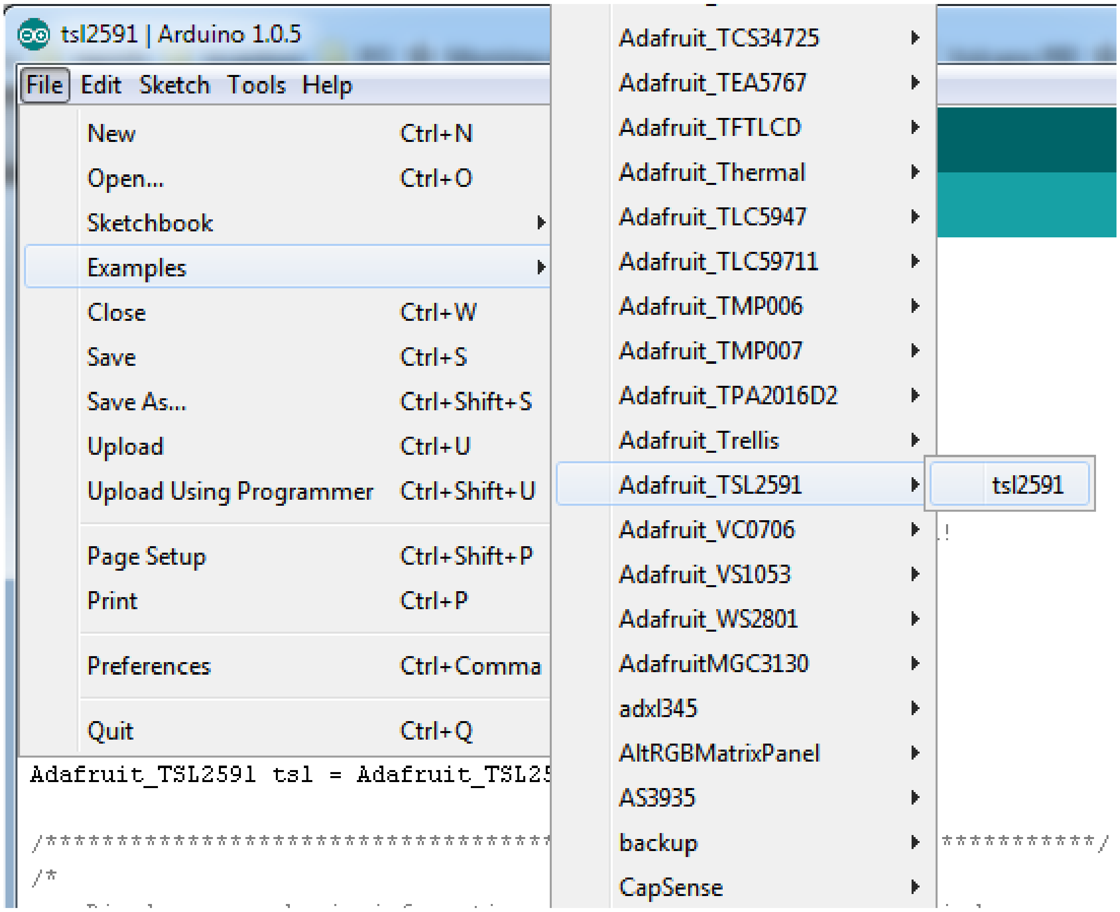
Download Adafruit_Sensor Library

http://adafru.it/cMO

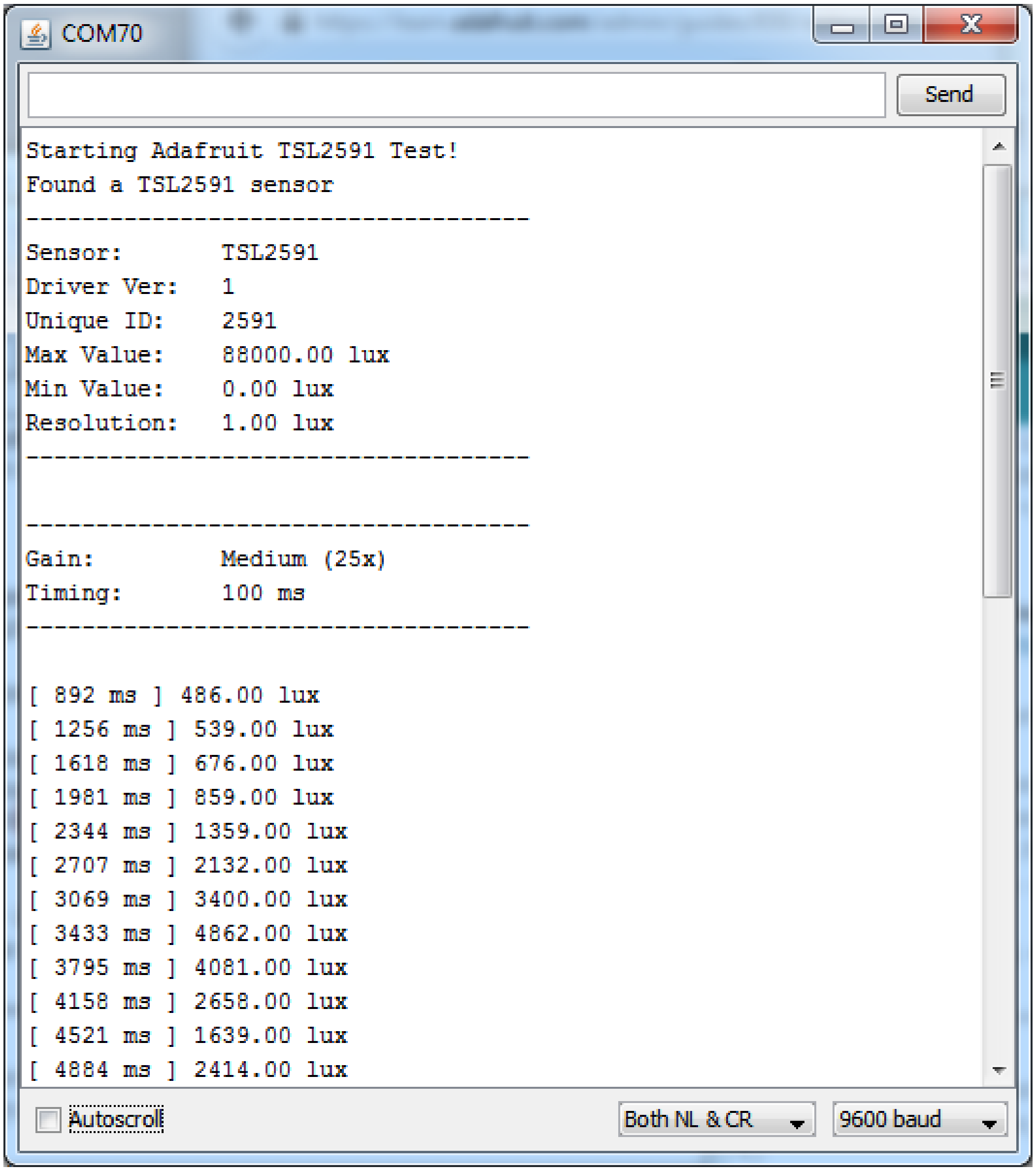
Install like you did with Adafruit_TSL2591

Load Demo

Open up **File->Examples->Adafruit_TSL2591->tsl2591** and upload to your Arduino wired up to the sensor



Thats it! Now open up the serial terminal window at 9600 speed to begin the test.



Try covering with your hand or shining a lamp onto the sensor to experiment with the light levels!

Library Reference

The **Adafruit_TSL2591** library contains a number of public functions to help you get started with this sensor.

Constructor

To create an instance of the Adafruit_TSL2591 driver, simple declare an appropriate object, along with a 32-bit numeric value to identify this sensor (in case you have several TSL2591s and want to track them separately in a logging system).

```
Adafruit_TSL2591 tsl = Adafruit_TSL2591(2591);
```

Gain and Timing

You can adjust the gain settings and integration time of the sensor to make it more or less sensitive to light, depending on the environment where the sensor is being used.

The gain can be set to one of the following values (though the last value, MAX, has limited use in the real world given the extreme amount of gain applied):

- **TSL2591_GAIN_LOW**: Sets the gain to 1x (bright light)
- **TSL2591_GAIN_MEDIUM**: Sets the gain to 25x (general purpose)
- **TSL2591_GAIN_HIGH**: Sets the gain to 428x (low light)
- **TSL2591_GAIN_MAX**: Sets the gain to 9876x (extremely low light)

Gain can be read or set via the following functions:

- **void setGain(tsl2591Gain_t gain);**
- **tsl2591Gain_t getGain();**

The integration time can be set between 100 and 600ms, and the longer the integration time the more light the sensor is able to integrate, making it more sensitive in low light the longer the integration time. The following values can be used:

- **TSL2591_INTEGRATIONTIME_100MS**
- **TSL2591_INTEGRATIONTIME_200MS**
- **TSL2591_INTEGRATIONTIME_300MS**
- **TSL2591_INTEGRATIONTIME_400MS**
- **TSL2591_INTEGRATIONTIME_500MS**
- **TSL2591_INTEGRATIONTIME_600MS**

The integration time can be read or set via the following functions:

- **void setTiming (tsl2591IntegrationTime_t integration);**
- **tsl2591IntegrationTime_t getTiming();**

An example showing how these functions are used can be seen in the code below:

```
/**************************************************************************/
/*
    Configures the gain and integration time for the TSL2561
*/
/**************************************************************************/
void configureSensor(void)
{
  // You can change the gain on the fly, to adapt to brighter/dimmer light situations
  //tsl.setGain(TSL2591_GAIN_LOW);    // 1x gain (bright light)
  tsl.setGain(TSL2591_GAIN_MED);      // 25x gain
  //tsl.setGain(TSL2591_GAIN_HIGH);   // 428x gain
```

```
  // Changing the integration time gives you a longer time over which to sense light
  // longer timelines are slower, but are good in very low light situtations!
  tsl.setTiming(TSL2591_INTEGRATIONTIME_100MS);  // shortest integration time (bright light)
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_200MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_300MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_400MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_500MS);
  //tsl.setTiming(TSL2591_INTEGRATIONTIME_600MS);  // longest integration time (dim light)

  /* Display the gain and integration time for reference sake */
  Serial.println("------------------------------------");
  Serial.print  ("Gain:         ");
  tsl2591Gain_t gain = tsl.getGain();
  switch(gain)
  {
    case TSL2591_GAIN_LOW:
      Serial.println("1x (Low)");
      break;
    case TSL2591_GAIN_MED:
      Serial.println("25x (Medium)");
      break;
    case TSL2591_GAIN_HIGH:
      Serial.println("428x (High)");
      break;
    case TSL2591_GAIN_MAX:
      Serial.println("9876x (Max)");
      break;
  }
  Serial.print  ("Timing:       ");
  Serial.print((tsl.getTiming() + 1) * 100, DEC);
  Serial.println(" ms");
  Serial.println("------------------------------------");
  Serial.println("");
}
```

Unified Sensor API

The Adafruit_TSL2591 library makes use of the Adafruit unified sensor framework (http://adafru.it/dGB) to provide sensor data in a standardized format and scale. If you wish to make use of this framweork, the two key functions that you need to work with are **getEvent** and **getSensor**, as described below:

void getEvent(sensors_event_t*)

This function will read a single sample from the sensor and return it in a generic sensors_event_t object. To use this function, you simply pass in a sensors_event_t reference, which will be populated by the function, and then read the results, as shown in the following code:

```
/**************************************************************************/
/*
    Performs a read using the Adafruit Unified Sensor API.
*/
/**************************************************************************/
void unifiedSensorAPIRead(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  tsl.getEvent(&event);

  /* Display the results (light is measured in lux) */
  Serial.print("[ "); Serial.print(event.timestamp); Serial.print(" ms ] ");
  if ((event.light == 0) |
      (event.light > 4294966000.0) |
      (event.light <-4294966000.0))
  {
    /* If event.light = 0 lux the sensor is probably saturated */
    /* and no reliable data could be generated! */
    /* if event.light is +/- 4294967040 there was a float over/underflow */
    Serial.println("Invalid data (adjust gain or timing)");
  }
  else
  {
    Serial.print(event.light); Serial.println(" lux");
  }
}
```

Note that some checks need to be performed on the sensor data in case the sensor saturated. If saturation happens, please adjust the gain and integration time up or down to change the sensor's sensitivity and output range.

void getSensor(sensor_t*)

This function returns some basic information about the sensor, and operates in a similar fashion to getEvent. You pass in an empty sensor_t reference, which will be populated by this function, and we can then read the results and retrieve some key details about the sensor and driver, as shown in the code below:

```
/**************************************************************************/
/*
```

```
    Displays some basic information on this sensor from the unified
    sensor API sensor_t type (see Adafruit_Sensor for more information)
*/
/**************************************************************************/
void displaySensorDetails(void)
{
  sensor_t sensor;
  tsl.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" lux");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" lux");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" lux");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}
```

Raw Data Access API

If you don't wish to use the Unified Sensor API, you can access the raw data for this sensor via the following three functions:

- **uint16_t getLuminosity (uint8_t channel);**
- **uint32_t getFullLuminosity ( );**
- **uint32_t calculateLux (uint16_t ch0, uint16_t ch1);**

getLuminosity can be used to read either the visible spectrum light sensor, or the infrared light sensor. It will return the raw 16-bit sensor value for the specified channel, as shown in the code below:

```
/**************************************************************************/
/*
    Shows how to perform a basic read on visible, full spectrum or
    infrared light (returns raw 16-bit ADC values)
*/
/**************************************************************************/
void simpleRead(void)
{
  // Simple data read example. Just read the infrared, fullspecrtrum diode
  // or 'visible' (difference between the two) channels.
  // This can take 100-600 milliseconds! Uncomment whichever of the following you want to read
  uint16_t x = tsl.getLuminosity(TSL2591_VISIBLE);
  //uint16_t x = tsl.getLuminosity(TSL2561_FULLSPECTRUM);
```

```
  //uint16_t x = tsl.getLuminosity(TSL2561_INFRARED);

  Serial.print("[ "); Serial.print(millis()); Serial.print(" ms ] ");
  Serial.print("Luminosity: ");
  Serial.println(x, DEC);
}
```

getFullLuminosity reads both the IR and full spectrum sensors at the same time to allow tigher correlation between the values, and then separates them in SW. The function returns a 32-bit value which needs to be split into two 16-bit values, as shown in the code below:

```
/**************************************************************************/
/*
    Show how to read IR and Full Spectrum at once and convert to lux
*/
/**************************************************************************/
void advancedRead(void)
{
  // More advanced data read example. Read 32 bits with top 16 bits IR, bottom 16 bits full spectrum
  // That way you can do whatever math and comparisons you want!
  uint32_t lum = tsl.getFullLuminosity();
  uint16_t ir, full;
  ir = lum >> 16;
  full = lum & 0xFFFF;
  Serial.print("[ "); Serial.print(millis()); Serial.print(" ms ] ");
  Serial.print("IR: "); Serial.print(ir);  Serial.print("  ");
  Serial.print("Full: "); Serial.print(full); Serial.print("  ");
  Serial.print("Visible: "); Serial.print(full - ir); Serial.print("  ");
  Serial.print("Lux: "); Serial.println(tsl.calculateLux(full, ir));
}
```

calculateLux can be used to take both the infrared and visible spectrum sensor data and roughly correlate with the equivalent SI lux value, based on a formula from the silicon vendor that takes into account the sensor properties and the integration time and gain settings of the device.

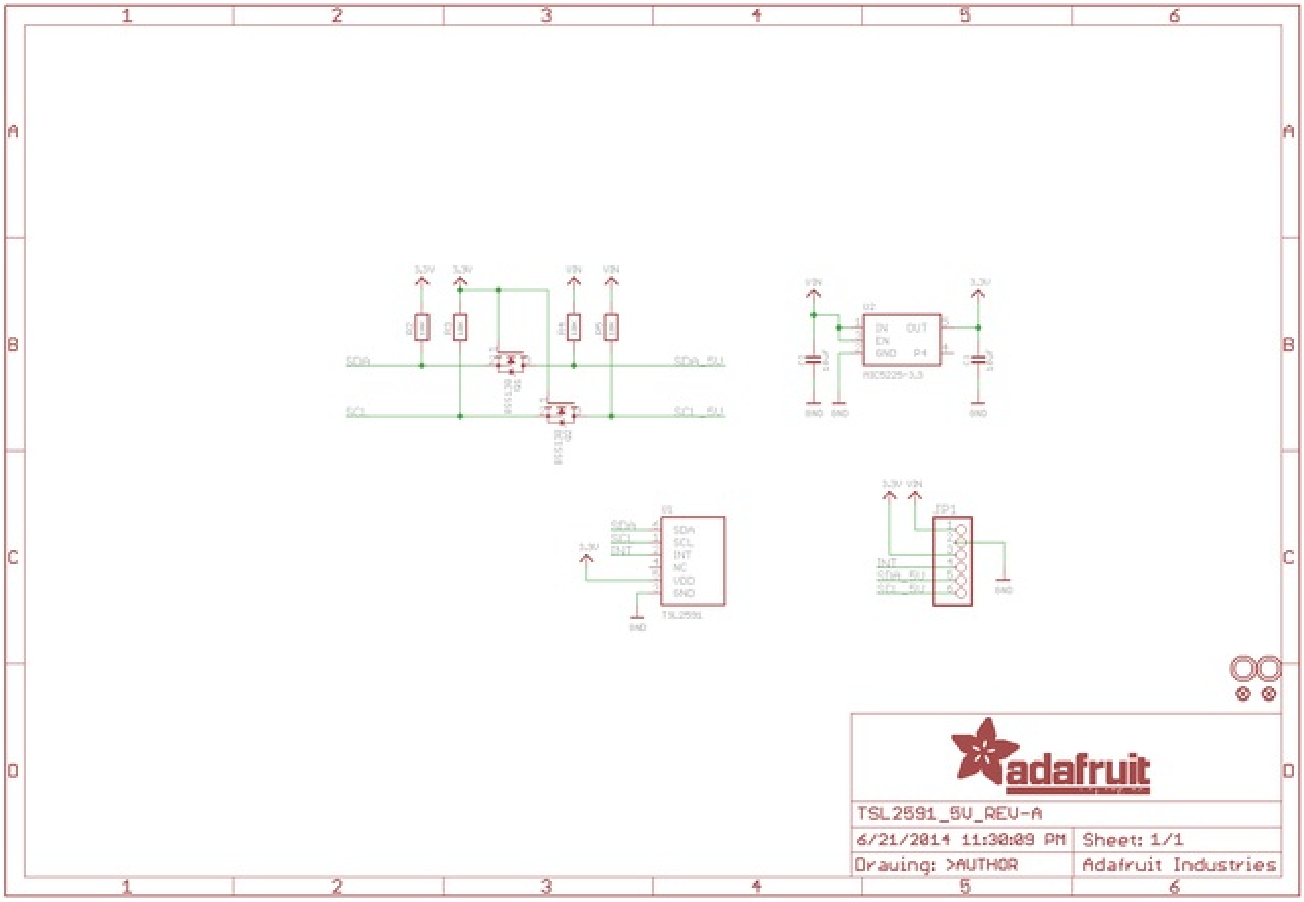
To calculate the lux, simple call **calculateLux(full, ir)**, where 'full' and 'ir' are raw 16-bit values taken from one of the two raw data functions above. See the code sample above for an example of calculating lux.

Downloads

Datasheets

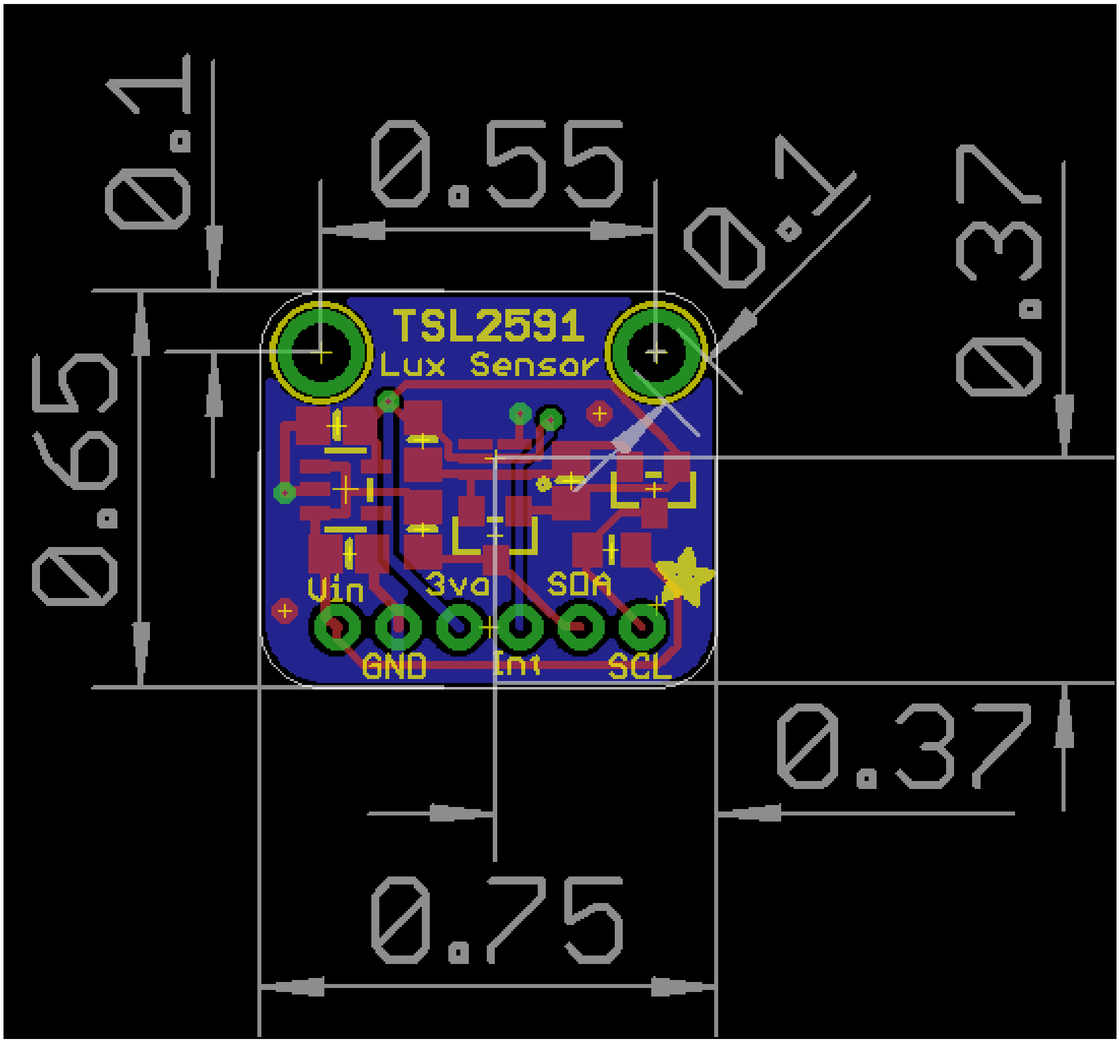
- TSL2591 Datasheet (http://adafru.it/dGs)

Schematic



Layout

(Dimensions are in Inches)


0.1
0.55
0.1
0.37
TSL2591
Lux Sensor
0.65
Vin
3vo
SDA
GND
Int
SCL
0.37
0.75