Application Note: DN40-Rev 1.0 – Lux and CCT Calculations using ams Color Sensors

DN40

Lux and CCT Calculations using ams Color Sensors

Table of Contents

Revision History

Revision	Date	Owner	Description
1.0	26.08.2013	H. Burton	Designer's Notebook – Lux and CC Calculation

1 General Description

Several **ams** digital color sensors provide red, green, blue and clear channel outputs with an integrated IR filter over the RGBC channels. Clear refers to an open channel with no color filter over the sensor. One of the most common applications is to measure the Lux and color temperature (CT) of the ambient light. This document will show how to calculate Lux and color temperature for **ams** color sensors with integrated IR filters. See the appendix to this app note for details on a specific part such as the TCS3x72.

In addition, there are many questions related to the implementation. What is the maximum lux level that can be measured? What is the accuracy of the lux calculations due to the digital nature of the measurements? Is the light source too bright resulting in saturation of the device? What happens when the device is placed behind dark glass? These are a few questions that are answered in this issue of the designer's notebook.

For information regarding the TSL2x7x, please see *DN29A Using the Lux Equation*.

2 Naming Convention

It is assumed that the reader is familiar with the **ams** color sensor data sheets. The naming convention in this document follows the data sheet, and the examples follow C coding conventions where words in capital letters represent variable names.

> ATIME[1] is the actual value programmed into the device
> ATIME_ms = (256 - ATIME) * 2.4; // Value converted to milliseconds (could be 2.8 for some devices)

Likewise for the AGAIN:

> AGAIN[2] is the two bit value programmed into the device to select gain
> AGAINx is the lookup table entry from AGAIN showing the amount of gain
> AGAINx will be 1,4,16 or 60 (or 64 for some devices)

The RGBC channels are red (RDATA), green (GDATA), blue (BDATA) and clear (CDATA) and are 16 bit values. In this document, RDATA will be abbreviated as R, GDATA as G, BDATA as B, and CDATA as C.

[1] RGBC Integration Time

[2] RGBC Gain Control

3 Calculations

3.1 IR Rejection

For some applications, the IR content is negligible and can be ignored. An example would be measuring the color temperature of an LED backlight. However, in applications that need to measure ambient light levels, incandescent lights and sunlight have strong IR contents. Most IR filters are imperfect and allow small amounts of residual IR to pass through. For IR intensive light sources, additional calculations are needed to remove the residual IR component.

In the TSL2x7x, the amount of IR light is related to the CH1 measurements. However, in the color sensors, there is no direct IR channel and the IR content must be calculated indirectly.

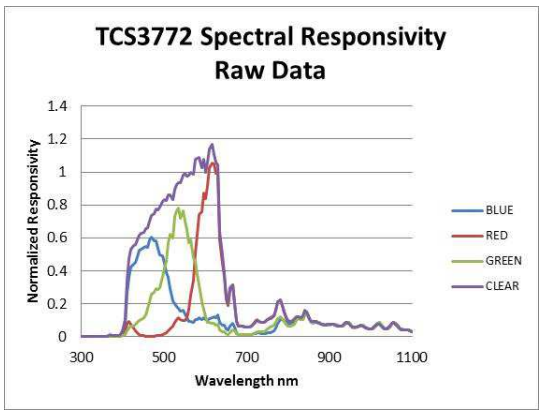

Figure 1a

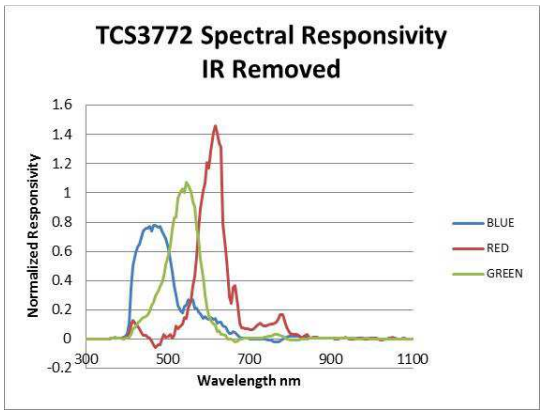

Figure 1b

TCS3772 Spectral Response – RAW Calculated Spectral Response – IR Removed

A few simple calculations can be used to remove the residual IR component of the light. Before we start the calculation, we will set some naming conventions. RGBC refers to the red, green, blue and clear channel data. The IR component can be calculated as follows:

$$IR = (R+G+B-C) / 2$$

The first level process is to remove the IR component. The IR compensated channels will be labeled as R', G', B', C'. Figure 1b shows an example of the calculated spectral response with the IR component removed.

$$R' = R – IR$$
$$G' = G – IR$$
$$B' = B – IR$$
$$C' = C – IR$$

3.2 Lux Calculations

The color Lux equation is a function of the R', G' and B' channels and associated color coefficients creating a G" as follows:

G" = R_Coef * R' + G_Coef * G' + B_Coef * B'

Note that the coefficient for the G' channel is unity. All calculation will hold this to unity and let the Counts per Lux (CPL)[3] be used to control the overall system gain. G" is directly related to lux by a factor that is a function of both integration time and gain. This is the same concept as with the TSL2x7x device as discussed in DN29A. Also note that Glass Attenuation (GA) * Device Factor (DF) may be combined and referred to as the Device and Glass Factor (DGF).

CPL = (ATIME_ms * AGAINx) / (GA * DF) = (ATIME_ms * AGAINx)/DGF

Lux = G" / CPL

3.3 Glass Attenuation Factor (GA)

For many applications, the ams device will be placed behind glass or plastic. Throughout this document, the term glass will refer to glass, plastic or other semitransparent material placed over the device. In many of these applications, a Glass Attenuation (GA) factor can be added to compensate for the lower light level at the device. However, if the glass or plastic has too much spectral distortion, a new lux equation may need to be generated.

The GA factor is intended to compensate for reduced light conditions. The GA factor is inversely proportional to the glass transmissivity (T). So, GA is defined as follows:

GA = 1/T.

For example, if the light is attenuated by 50%, then the glass is 50% transmissive, and the GA factor is 2 (1/0.5). With light attenuated by 95%, the glass is 5% transmissive, and GA factor is 20 (1/0.05).

[3] DN29 "CPL = (ATIME_ms * AGAINx) / (GA * DF);"

3.4 CT Calculations

Color temperature is related to the color with which a piece of metal glows when heated to a particular temperature and is typically stated in terms of degrees Kelvin. The color temperature goes from red at lower temperatures to blue at higher temperatures. The following shows a chart of color vs. temperature.

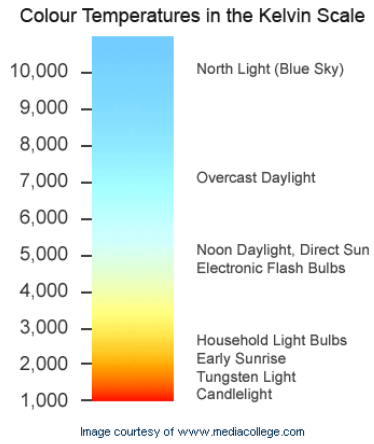

Figure 3. Color Temperature

A simple method to calculate color temperature (CT) is to use the ratio of blue to red light and use an empirical formula to determine the color temperature. As with the lux equation, IR cancellation is critical to the operation. This formula for CT is as follows:

$$CT = CT\_Coef * B' / R' + CT\_Offset$$

3.5 Device Count Saturation

As the light becomes brighter, the clear channel will tend to saturate first. When the clear channel saturates, the IR calculation algorithm breaks down and will no longer work. In this situation either the calculations need to be declared invalid, or a more complex algorithm can be implemented to extend the sensing range. Since R+G+B are approximately equal to C, the clear channel will tend to saturate much faster than the RGB channels. See the section below on “Extending the Sensitivity Range”.

Before calculating lux, it is important to understand device saturation. There are two conditions for device saturation: analog saturation and digital saturation. Analog saturation is when the analog input is greater than what can be accumulated with the light-to-frequency conversion. Digital saturation is when the digital accumulator is overflowing before the analog saturates.

The full scale value for analog saturation depends upon the integration time programmed into the device. In saturation, the device accumulates 1024 counts for each 2.4 ms (in many devices) of integration time up to a maximum of 65,535 counts. Analog saturation will occur up to an integration time of 154 ms.

If the ALS integration time is greater than 154 ms (ATIMEx ≤ 64), digital saturation will occur before analog saturation. Digital saturation occurs when the count reaches 65,535.

```
if ( (256 – ATIME)  >  63)          // if ATIME_ms > 154ms (Digital Saturation)
        SATURATION = 65535;
else                                // if ATIME_ms <= 154ms (Analog Saturation)
        SATURATION = 1024 * (256 − ATIME);
```

3.6 Ripple Rejection

One of the first factors impacting the integration time decision is 50/60Hz ripple rejection. If the programmed integration time is in multiples of 10 ms and 8.3 ms (the half cycle time), both frequencies are rejected. An integration time value of 50ms or multiples of 50ms are required to reject both 50Hz and 60Hz ripple. In cases requiring faster sampling time, averaging over a 50ms period may be needed to reject the fluorescent light and incandescent light ripple.

3.7 Ripple Saturation

Ripple saturation is a second condition that impacts saturation. If there is ripple in the received signal, then the signal will fluctuate in and out of saturation, and the value read from C will be less than the maximum but still have some effects of being saturated. Under this condition, a channel reading may be slightly below the saturated calculation but in reality be saturated during the peaks, resulting in a value less than the actual light level. At integration times > 150ms, digital saturation occurs before the analog saturation; therefore, this calculation would not be necessary. The following shows the 75% calculations using integer math.

```
SATURATION75 = SATURATION – SATURATION / 4; // if ATIME_ms < 150
```

The saturation check should be done before the lux calculation and should return an overflow code if the device is saturated. Saturation is checked only against C since the C is always > (R or G or B). A saturation check is also useful in several other situations including autogain and extending the lux range. See the sections below for more details.

3.8 Light Sources

Using a ratio of the IR content to the clear channel, a CRATIO can be developed which can be used to determine the type of light is present. Note the ratio should always be less than 1. Also note the ratio should only be calculated when the clear channel is not saturated. For fluorescent light, the CRATIO is low. For sunlight, CRATIO is a medium value. For incandescent light, CRATIO is a high value. When the amount of IR is very low, e.g. in the case of an LED or fluorescent light source, the ratio may vary depending upon the specific device. However, if the maximum ratio is always held, it can be assumed that this ratio is an incandescent light. Sunlight will be about ½ of this value and Fluorescent/LED light will be very close to zero (< 0.1). Tests of limited parts show that incandescent parts can have a ratio around 0.3.

3.9 Maximum Lux value

To calculate maximum lux, several simplifications will be taken. If we take that G" = G', then

Lux = G' / CPL = (G – IR) / CPL
Lux = (G – 0.5* R - 0.5* G - 0.5* B + 0.5* C)/ CPL;
Lux = (0.5* G – 0.5* R - 0.5* B + 0.5* C)/ CPL;
Lux = (G – R – B + C) / (CPL * 2);

If G=R=B=C/3 (a very rough estimation for white light) then:

Lux = (C/3 – C/3 – C/3 + C) / (CPL * 2);
Lux = G / CPL;
Lux = C / (CPL * 3);

This can help for easy testing

MaxLux = 65k / (CPL * 3);

3.10 Lux Accuracy

A simple formula can be used to determine the lux accuracy due to the Digitization Error (DER) present in the data increments by integer amounts. The following formula is used to calculate the lux accuracy:

DER = (+/- 2) / CPL

It is recommended that CPL be greater than 5 to have a lux accuracy < +/- 0.5 lux.

3.11 Equalization

The next step is optional depending upon the application and provides a more accurate result for the color saturation calculations (see below). This step equalizes the response of the RGB channels. The process is a matrix multiply but can be simply stated as follows:

$$\begin{bmatrix} c00 & c01 & c02 \\ c10 & c11 & c12 \\ c20 & c21 & c22 \end{bmatrix} \quad X \quad \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \quad = \quad \begin{bmatrix} R'' \\ G'' \\ B'' \end{bmatrix}$$

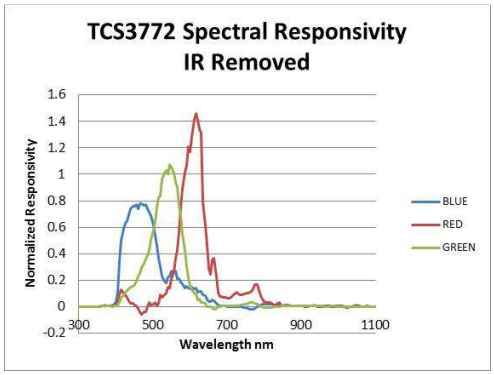

Figure 2a
Calculated Spectral Response
IR Removed

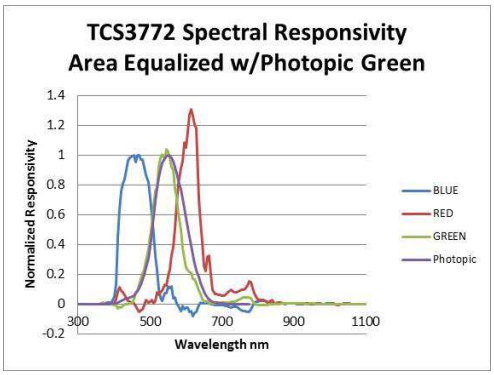

Figure 2b
Calculated Spectral Response
RGB Areas Equalized w/Photopic Green

The matrix starts with a c00 notation to correlate with the matrix indices used in programming. Note that in the equalization matrix, the equalized green channel should be a photopic channel where:

c10 = R_Coef
c11 = G_Coef
c12 = B_Coef

Also note that C11 or G_Coef is forced to unity.

3.12 Color Saturation

As the color becomes saturated, the lux (and CT) estimates become less accurate. To determine when this happens, a calculation of color saturation can be used to determine this condition and then the CT results can be ignored or other methods of calculation can be utilized. The following can be used to determine saturation:

M = max (R”, G”, B”)
m = min (R”, G”, B”)
Saturation = (M – m) / M

For white light, R”~G”~B” and M – m is small and (M – m) / M is smaller. For saturated light, M – m is large and if (M – m) / M > 0.75, the light source is starting to saturate.

Color saturation calculation may be the only use for the transformed matrix using R” and B”. If this is not needed, then this calculation along with the matrix multiply may not be needed.

3.13 Extending the Sensitivity Range

If higher lux reading are required, an algorithm can be implemented which will extend the lux range up to 3x. This algorithm will work with the assumption that the light source type does not change.

To implement this algorithm, the system should monitor the 75% saturation point and save the RGB values at that point and set a flag indicating that the saturation algorithm should be utilized. In this case, the system should save the RGB values as:

Rx = R (known good data)
Gx = G (known good data)
Bx = B (known good data)
IRx = IR (known good data)

The ratios of IR to each color are defined as:

Rratio = IRx / Rx
Gratio = IRx / Gx
Bratio = IRx / Bx

Then the calculation for R' G' B' becomes:

R' = R - R * Rratio
G' = G - G * Gratio
B' = B - B * Bratio

We can then replace the Rratio with an IR factor (R_IRF).

R_IRF = 1 – Rratio
G_IRF = 1 – Gratio
B_IRF = 1 – Bratio

Then the calculation for R' G' B' becomes:

R' = R * R_IRF
G' = G * G_IRF
B' = B * B_IRF

3.14 Low Count Level and Autogain

When the counts get very low, the accuracy of the lux calculation will also be diminished. For example in very low light with no IR light present the following could be true:

IR = (R+G+B-C) / 2
IR = (1+1+1-3) / 2 = 0
Lux = 0 .136 + 1 – .444 = 0.9 lux //assume CPL = 1

However, if there is noise or minor light changes such that B or G goes to 2, then this becomes

IR = (1+1+2-3) / 2 = 0.5
R= 0.5, G= 0.5, B= 1.5
Lux = 0 .136 * 0.5 + 1 * 0.5 – .444 * 1.5
Lux = 0.068 + .5 - .666 = -.098 lux

IR = (1+2+1-3) / 2 = 0.5
R=0.5, G= 1.5, B=0.5
Lux = 0 .136 * 0.5 + 1 * 1.5 – .444 * 0.5
Lux = 0.068 + 1.5 - .222 = 1.346

This causes a large fluctuation with a very small count change. It is recommended that the green channel counts be above 10 to ensure accuracy for the lux calculation. Also, this can allow the lux to go slightly negative. For this reason, the lux equation is a signed integer that tests for negative values and returns a zero when a negative value is calculated.

A recommended option for solving this problem is to implement an autogain algorithm such that when the clear channel count drops below 100, an autogain algorithm is used to increase the gain. See DN39 for more details on an autogain algorithm.

3.15 Efficient Integer Calculations

The calculations to remove IR are very simple and only require shifts and adds.

IR = (R+G+B-C) >> 1
R’ = R – IR
B’ = B - IR
G’ = G – IR

Lux calculations are also simple but have fractional coefficients. One trick is to multiply the coefficients by 1000 making them integer values and using ATIME in microseconds (also an integer) to cancel out this multiplication.

Gi” = 136 * R’+ 1000 * G’ +(-444) * B’

Note that CPL is calculated only when the ATIME or AGAIN has changed. This is why it is shown as a separate calculation.

CPL = (ATIME_us * AGAINx) / DGF

Lux = Gi" / CPL

For color temperature, make sure the CT_Coef value is multiplied by B' before doing the divide by R '.

CT = (CT_Coef * B') / R' + CT_Offset

For the ratios, multiply IRx by 1000 before the divide.

Gratio = IRx * 1000 / Gx

3.16 Dark Glass or Ink on Glass

If ink is used to hide the sensor, all of the above calculations must be recomputed. Dark glass or ink on the glass will typically alter the spectrum. Typically, this will allow more blue light and IR light to be transmitted, while attenuating the green and red light. This shifts the spectrum and requires recalculation of all of the equation coefficients.

APPENDIX I: Coefficients for Lux and CT Equations

Equation Summary:

Lux = (R_Coef * R' + G_Coef * G' + B_Coef * B') / CPL, where:
CPL = (AGAINx * ATIME_ms) / (GA * DF), and
R' = R – IR, G' = G – IR, B' = B – IR, where IR = (R + G + B – C)/2.
CT (degrees Kelvin) = CT_Coef*(B'/R') + CT_Offset

Table 1: Coefficients for Lux and CT equations. For parts operating in open air, with GA = 1.

Device	GA*	DF	R_Coef	G_Coef	B_Coef	CT_Coef	CT_Offset
TCS3414	1.0	127	-0.097	1.000	-0.482	3852	1855
TCS3472	1.0	310	0.136	1.000	-0.444	3810	1391
TCS3772	1.0	310	0.136	1.000	-0.444	3810	1391
TMD3782	1.0	312	0.093	1.000	-0.522	4916	1427
TCS3790	1.0	385	118	1.000	-415	4201	1495
TMD3790	1.0	233	129	1.000	-378	3900	1568

*Placing the parts behind clear glass requires using GA = 1.08 in the formula for CPL above, to account for Fresnel reflection at the glass surfaces. Placing the parts behind an aperture may require adjustment of the GA coefficient to account for attenuation of the light by the aperture.

Restating the previous data in integer format by multiplying the coefficients by 1000 and combining the GA and DF into a single DGF term creates the following table:

Device	DGF	R_Coef	G_Coef	B_Coef	CT_Coef	CT_Offset
TCS3414	127	-97	1000	-482	3852	1855
TCS3472	310	136	1000	-444	3810	1391
TCS3772	310	136	1000	-444	3810	1391
TMD3782	312	93	1000	-522	4916	1427
TCS3790	385	118	1000	-415	4201	1495
TMD3790	233	129	1000	-378	3900	1568

IMPORTANT: Placing the parts behind dark glass will likely require custom calculation of the coefficients, as most dark glass is not spectrally neutral and every manufacturer's glass and ink is unique the their specific make and model of product.

APPENDIX II: Data Structures

Data Structures

The following are suggested data structures that can be used with the device. The all are in integer format. If the name is followed by a 100, it is the value times 100 and is typically used when calculating percentages. If the number is followed by 1000, it is the actual number times 1000.

The first structure is a structure of structures. This provides a logical way of organizing the data.

```
// Group of structures organized by function
public struct s_ams_Color_Chip
{
    public s_ams_APP_Color_Setup APP_Color_Setup;
    public s_ams_APP_Color_Data APP_Color_Data;
    public s_ams_APP_Callback Callback;
    public s_ams_Device Device;
}
```

The color setup data is used during initialization to set operational parameters of the device.

```
// Sets up function that impact only the color/ALS functions
public struct s_ams_APP_Color_Setup
{
    public bool Auto_Gain_Enable;       // Enables gain to be adjusted automatically
    public bool Lux_Extend_Enable;      // Enables lux extend to sense brighter light //
                                        levels
    public int Clear_Threshold_Hi_100;  // Hi percentage for upper threshold
    public int Clear_Threshold_Lo_100;  // Lo percentage for lower threshold
    public int DGF;                     // Device and Glass Factor – See DN
    public int CT_Coef;                 // Color temperature coefficient for CT
                                        // calculation
    public int CT_Offset;               // Color temperature offset for CT calculation
    public int[,] CalArray;             // 3x3 cal array [1,0]=R_Coef [1,1]=G_Coef
                                        // [1,2]=B_Coef
}
```

The color data function during each interrupt of the device.

```
// Color data calculate during each poll/interrupt cycle
public struct s_ams_APP_Color_Data //dynamic data that may change every data sample
{
    public DateTime timestamp;   //time when data was captured
    public bool Lux_OK;          // No I2C error or overflow error occurred in the calc
                                 // of lux
    public bool CT_OK;           // No I2C error or overflow error occurred in the calc
                                 // of CT
    public bool Lux_Extended;    // Lux extension algorithm was used in the last lux
                                 // calc
    public uint IR;              // IR content as (R+G+B-C)/2
    public iColor RGBi;          // Color without IR content
    public iColor RGBii_1000;    // Color after equalization matrix multiply * 1000
    public iColor IRF;           //Used with lux extension
    public int Lux_1000;         //output in millilux
    public int CT;               // color temperature output
    public uint Clear_Ratio100;  // indicate amount of IR light as a percentage * 100
    public uint Saturation100;   // percentage color saturation * 100
    public uint Color_Threshold; //Threshold used for last color event
    public bool Color_Interrupt_State_Hi; //Indictes if ALS interrupt was lux up or down
}
```

The callback functions are used to report events or for system debug.

```
// Callback function
public struct s_ams_APP_Callback //dynamic data that may change every data sample
{
    public Error_Handler Error_Callback; // Text output port for error reporting
```

```
    public Event_Handler Event_Callback; // Interrupt callback into
}
```

The the iColor structure stores RGB data.

```
public struct iColor
{
    public uint Red;
    public uint Green;
    public uint Blue;
}
```

The Device structure stores the current state of the device. This is directly related to the register value but is in human readable form where possible

```
public struct s_ams_Device // Information directly related to register values
{
    public byte Enable;          // Register value, 0 is device off
    public uint Wait_xTime;      // 256 – Register value
    public byte Wait_Long;       // Register bit (non zero indicates a 1)

    public byte ALS_xGain;       // Gain in human readable form 1, 4, 16 or 60
    public uint ALS_xTime;       // 256 – Register value

    public UInt16 AILT, AIHT;    // 16 bit register values
    public byte APers;           // 4 bits register sub-value

    public byte ID;              // Register value
    public byte Status;   // Register value
    public byte Config;          // Register value
    public byte Rev;

    public iColor Color;  // 16 bit register values
    public UInt16 Clear;  // 16 bit register
}
```