Application Report
DLPA063 – October 2015

Real-Time Color Management for DLPC343x — White Point Correction (WPC)/ Color Coordinate Adjustment (CCA)/ Content Adaptive Illumination Control (CAIC)

Tony Lee, Edward Chang FAE

ABSTRACT

This application report introduces a new WPC, CCA color management system by using a 3-color digital sensor on a DLPC343x-based LED projector. In addition to the new sensor hardware, the color management system requires a new software algorithm built on the MSP430 MCU and communicates in between the frond-end system and the DLPC343x device to make the real-time color management system work properly.

Contents

List of Figures

1 Introduction

The system provides an ease-of-use color management system with the following features:

- Real-time white point (color temperature) correction
- Monitoring color point LED illuminators in CIE1931 x, y, Y format
- Monitoring LED efficiency
- Adjust R, G, B, C, M, Y, W P7 register at run time
- Adjust hue, saturation, gain (HSG) for R, G, B, C, M, Y at run time
- Adjust x, y color point (CCA) for R, G, B, C, M, Y, W at run time
- Build CAIC current versus intensity table for R, G, B LED

The real-time color management description of new WPC, CCA, and CAIC will be referred to as *New WPC* throughout this application note.

2 System Installation

2.1 Sensor Installation in Optical Engine

The sensor is installed at the illumination path where it senses the stray light from the LED module as shown in Figure 1. In general, the existing sensor position can be used for a new WPC digital sensor. However, a common issue that may cause a sensor reading integrity problem is DMD off-state light interference. Figure 2 shows an example; the sensor senses the unwanted DMD off-state light (as indicated by red arrows) that result in new WPC performing incorrectly.

A simple test determines if the sensor position has off-state light interference. First, display a white pattern and read the sensor. Second, display a black pattern and read the sensor again. The qualified sensor position should have approximately the same readings on both test patterns. Otherwise, the sensor position has off-state light interference as described previously.

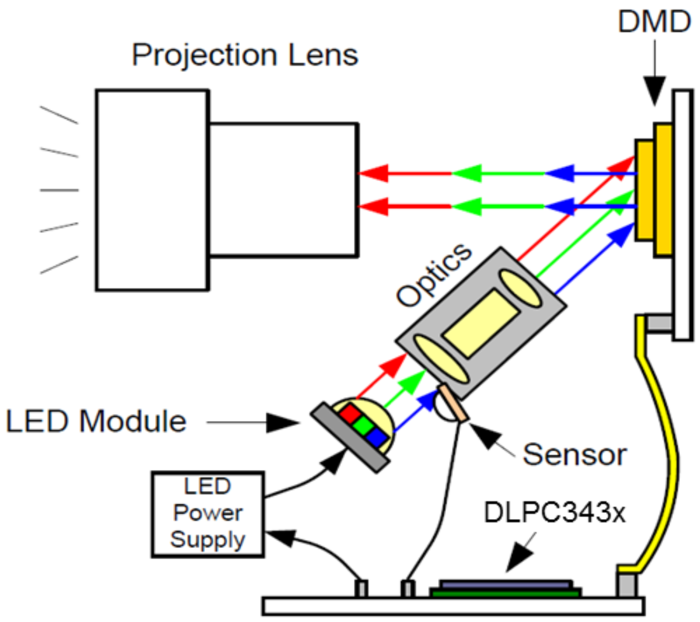


Figure 1. Sensor Position in Optical System

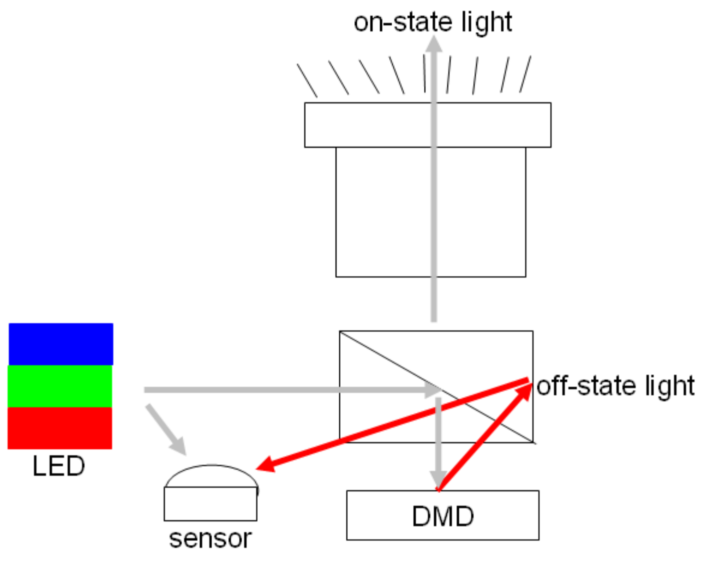


Figure 2. DMD Off-State Light Interference

2.2 Sensor Circuit

This application note uses the Capella digital sensor (www.capellamicro.com) to develop the New WPC algorithm. Capella CM3303 (package: 2.35 mm × 1.8 mm × 1 mm) has 6 pins out with an input range of 2.7 V to 5.5 V. The CM3303 has three individual R, G, and B sensors with a built-in ADC which outputs fixed pulse width at 170 ns. The higher the light intensity to the sensor, the higher the pulse frequency density. To implement the sensor circuit on MSP430 MCU, directly connect sensor output pins to MSP430 MCU GPIO pins. The GPIO pins are programmed as input ports with hardware interrupt trigger by the sensor output pulses. The sensor circuit does not require an ADC or any other devices for data acquisition, as shown in Figure 3.

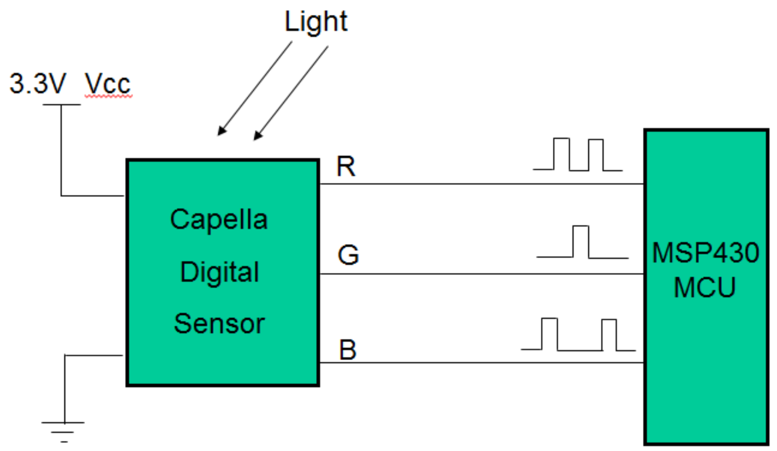


Figure 3. Sensor Circuit

2.3 New WPC System Block

In addition to a RGB color sensor, TI MSP430F5310 MCU must run the WPC algorithm to support New WPC. MSP430 is placed in between the frond-end processor and DLPC343x with I^{2}C communication protocol. MSP430 enables four hardware interrupts to monitor RGB sensor feedback and also Vsync. Vsync on MSP430 is routed to DLPC343x TSTPT_2 to monitor both internal and external sources. To enable Vsync on TSTPT_2 pin on DLPC343x needs the following two commands are added in autoinit.bf file.

```
W 36 e5 3c 20 00 40
W 36 e6 07 07 00 06
```

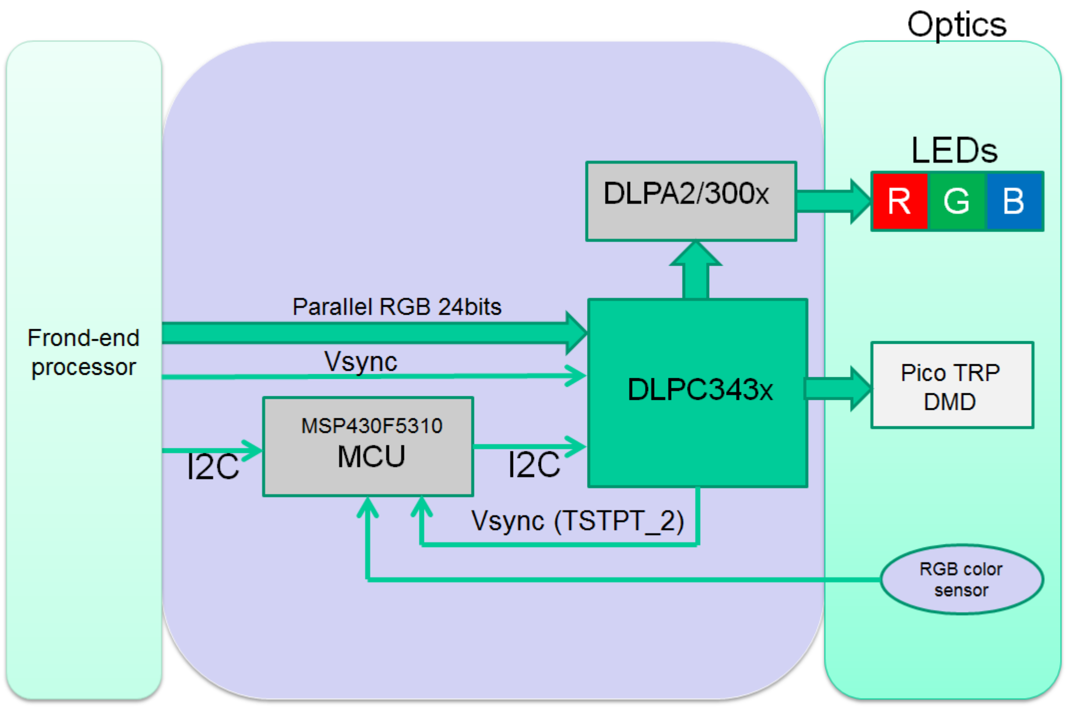


Figure 4. New WPC System Block Diagram

MSP430F5310 MCU has two I^{2}C buses as shown in Figure 5. One bus is configured as I^{2}C slave and connects to front-end CPU; another I^{2}C bus is configured as I^{2}C master to control DLPC343x. Two type of commands are sent from the frond-end CPU.

- **DLPC343x General Commands:** Frond-end CPU talks to MSP430. MSP430 repeats the same command to DLPC343x as either read or write on the I^{2}C master end. See the *DLPC3430, DLPC3435, DLPC3433, and DLPC3438 Software Programmer's Guide* (DLPU020) for a detailed command description.
- **New WPC Commands:** MSP430 accepts New WPC commands that start with 0xFF and use the same DLP device address (0x36) on I^{2}C slave end. See Table 46 for more information.

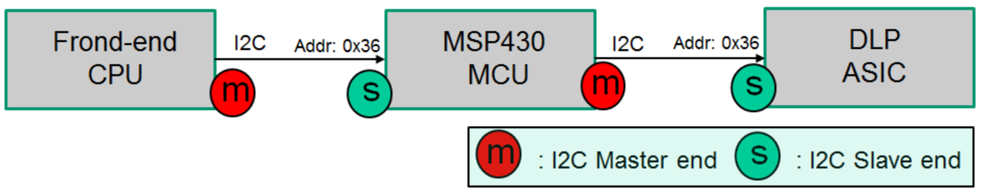


Figure 5. New WPC I^{2}C Buses Configuration

2.4 Reference Circuit With MSP430 MCU

Figure 6 shows the recommended schematic of MSP430F5310 and color sensor. This schematic is available in PDF format along with the application note on www.ti.com. The reference MSP430 software is developed based on the circuit. The Vsync (TSTPT_2) from DLPC343x is 1.8 V and MSP430 requires a 3.3-V signal and a level shifter.

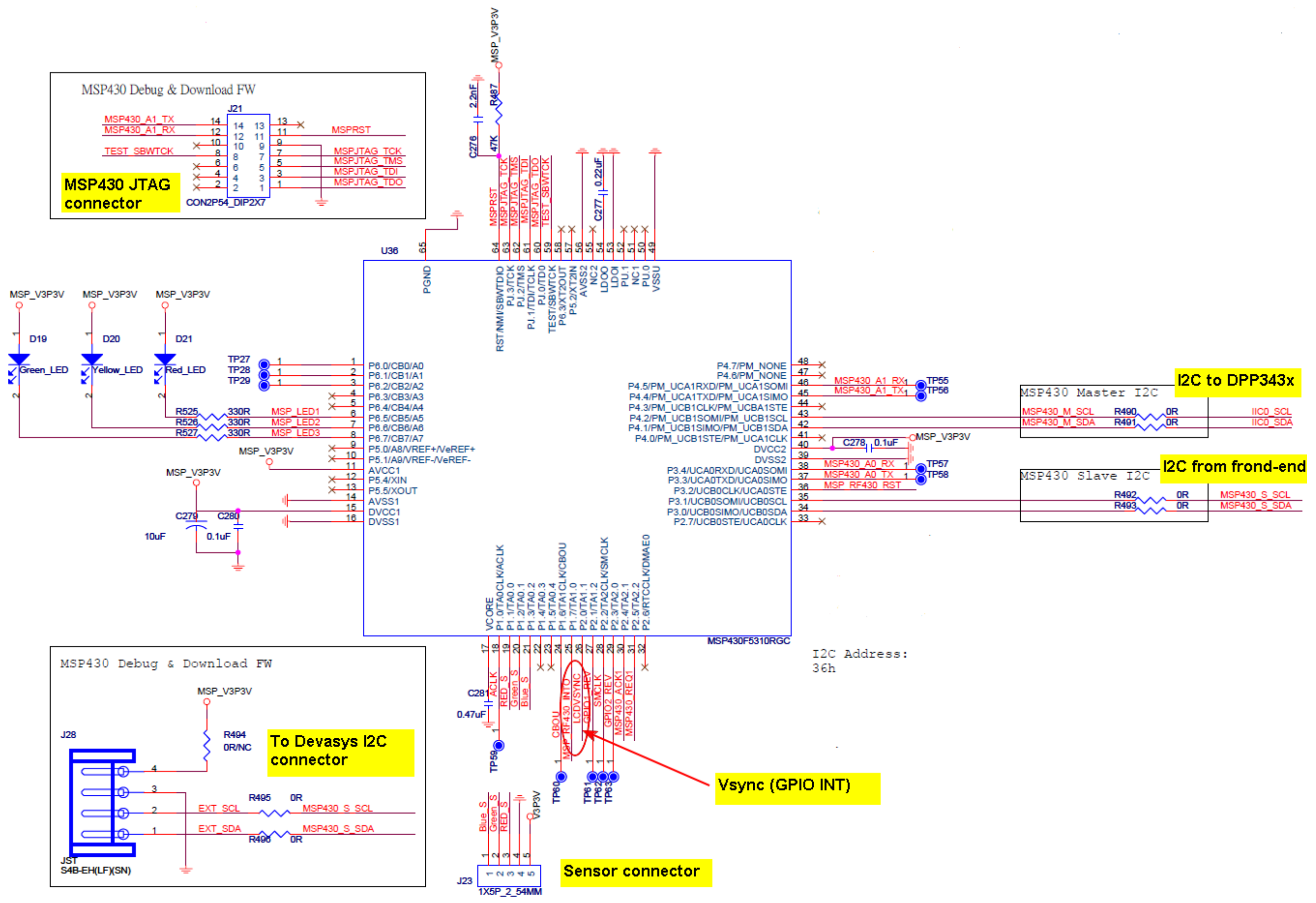
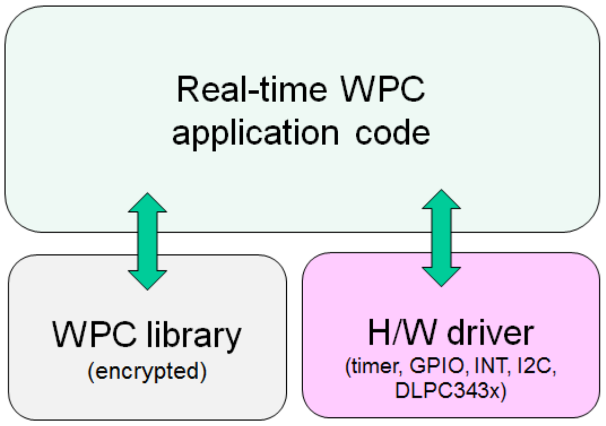


Figure 6. New WPC Schematic

3 Software Structure on MSP430

3.1 Main Process State Machine



Real-time WPC application code
WPC library
(encrypted)
H/W driver
(timer, GPIO, INT, I2C, DLPC343x)

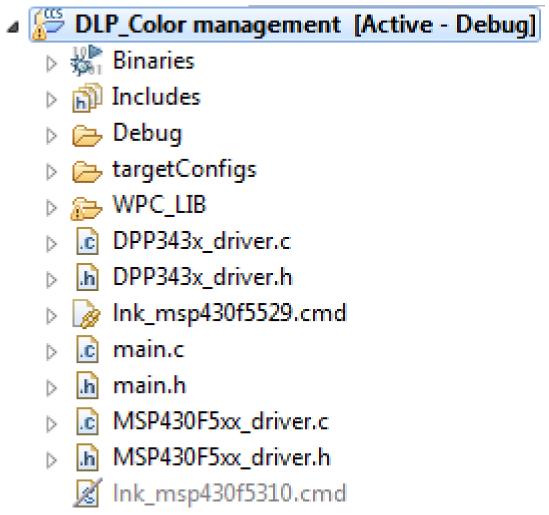
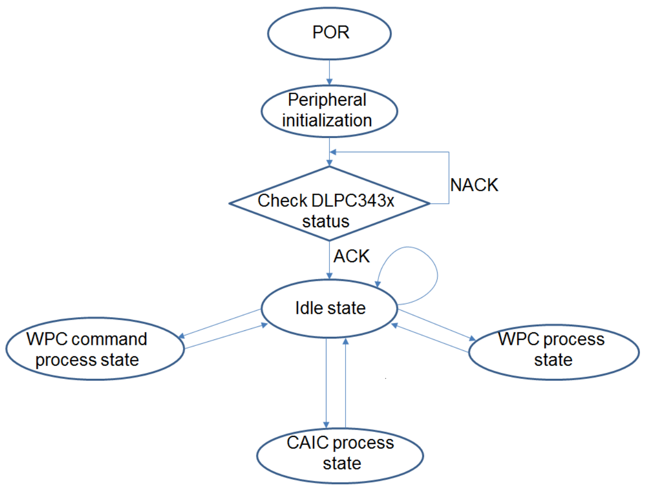

DLP_Color management [Active - Debug]
Binaries
Includes
Debug
targetConfigs
WPC_LIB
DPP343x_driver.c
DPP343x_driver.h
lnk_msp430f5529.cmd
main.c
main.h
MSP430F5xx_driver.c
MSP430F5xx_driver.h
lnk_msp430f5310.cmd

Figure 7. MSP430 Code Structure

The three parts in the project are the application code, hardware driver, and encrypted WPC library. The application code manages the code state machines, which are Idle state, Command Process state, WPC state, and CAIC Build state. In the idle state, it constantly check whether or not the WPC command is received or the CAIC build status is in progress. Users can add custom code here or add new states for extra features. The WPC process state does a calculation for a time frame defined by WPC_FREQUENCY. Note that the added code must finish its process before the next frame time starts. The subroutine MainStateChange is used to check the criterion of exiting the current state and entering the next state. This helps users to manage the state machine and also the ease of debugging.



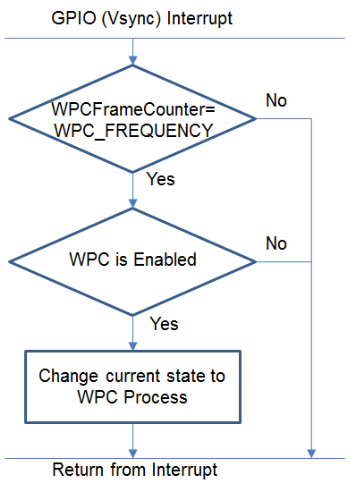


Figure 8. Software State Machine Flowchart

3.2 I²C Command Process Flow

The MSP430 requires two hardware I²C buses to receive commands from the front-end video processor and send WPC commands to DLPC343x. Figure 9 shows the block diagram.

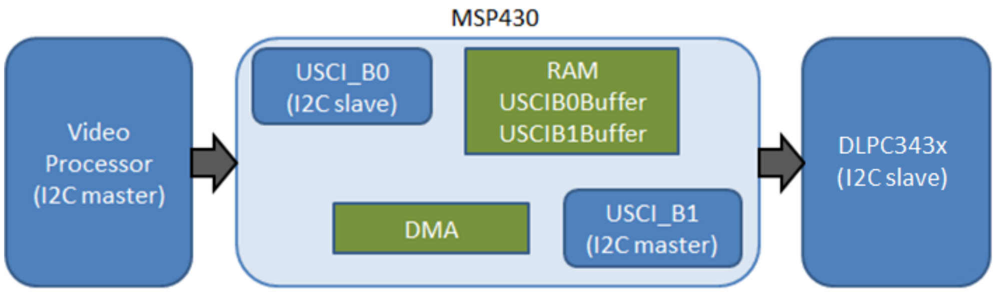


Figure 9. I²C Configuration

The three possible scenarios are:

1. The video processor sends data to DLPC343x: All the I²C data from the video processor is directly stored in USCIB0Buffer using DMA. After the USCI_B0 module receives an I²C stop signal, the entire data will be transferred to USCI_B1 module through DMA channel if the start byte is not 0xFF. See Figure 10.

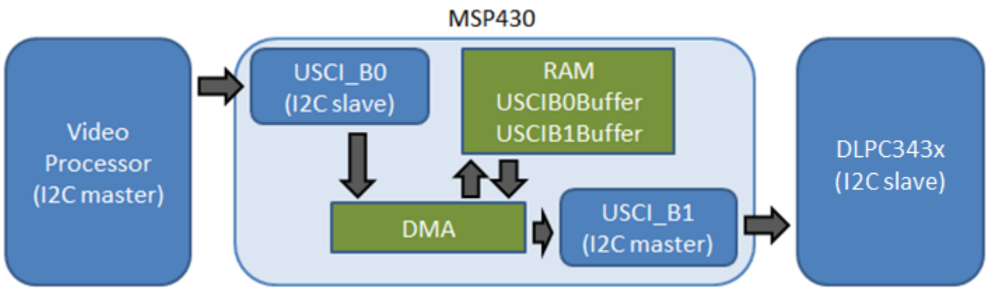


Figure 10. I²C Bypass Mode

2. The video processor sends a WPC command to MSP430. The data is stored in the USCIB0Buffer array and sets the flag to enter WPC command process state. See Figure 11.

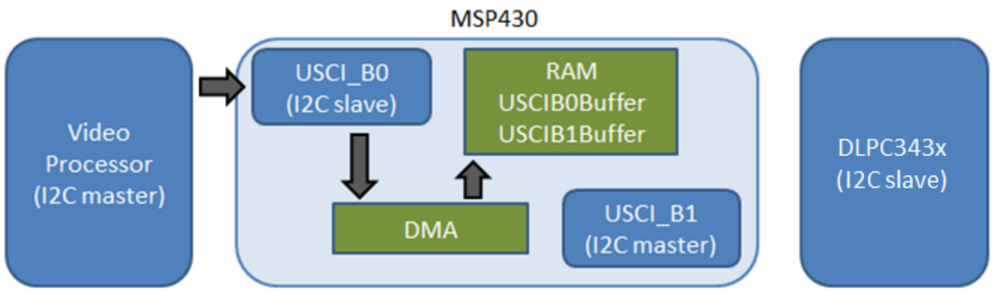


Figure 11. MSP430 Receives WPC Command from the Video Processor

3. MSP430 sends a command to DLPC343x. Both of the I²C read and write commands are first stored in the USCIB1Buffer array, and then trigger DMA to transfer data to USCI_B1 module automatically. See Figure 12.

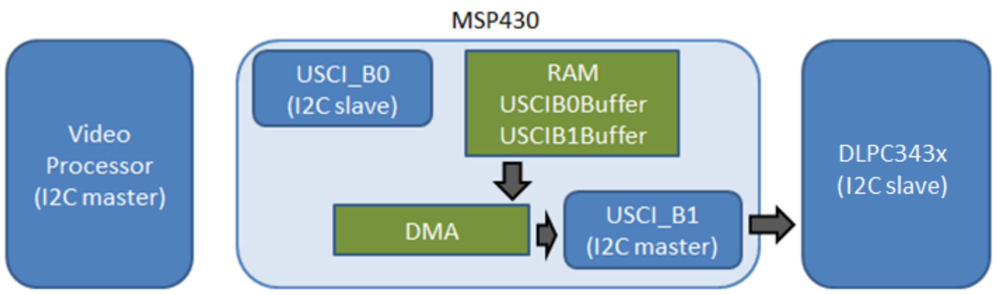


Figure 12. MSP430 Sends Command to DLPC343x

4 New WPC Work Principle

4.1 WPC

The digital sensor is configured to monitor the whole frame time by Vsync and readjusts the sense time whenever Vsync frequency changes. Thus, the sense time always matches the audience's perception; Figure 13 shows an example.

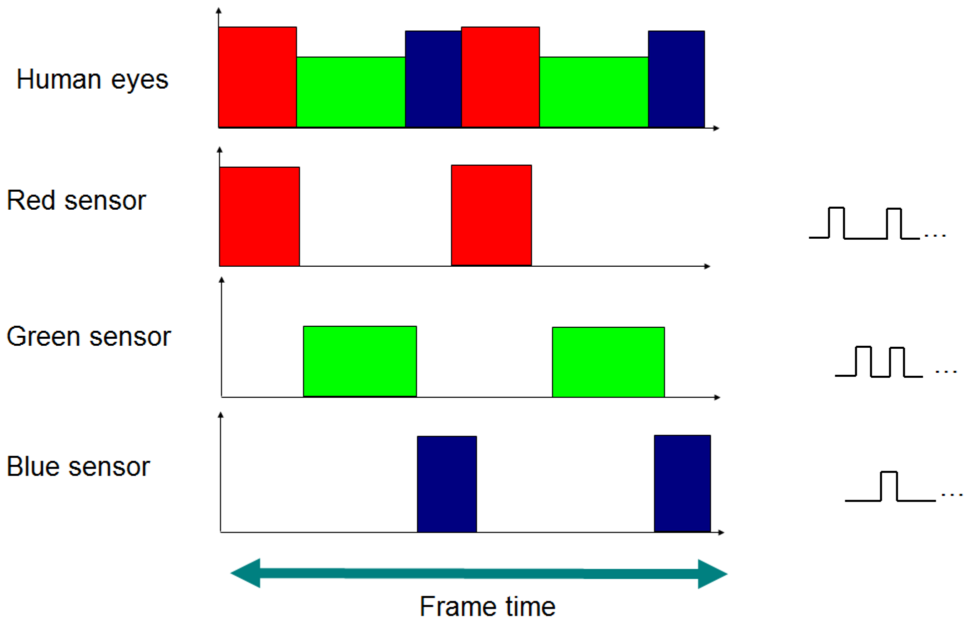


Figure 13. Sensor Monitor the Whole Frame Time

While the R, G, B sensor feedback pulse counts to MSP430 in frame-basis, the new WPC algorithm proceeds two steps after enabling new WPC or changing the desired white point.

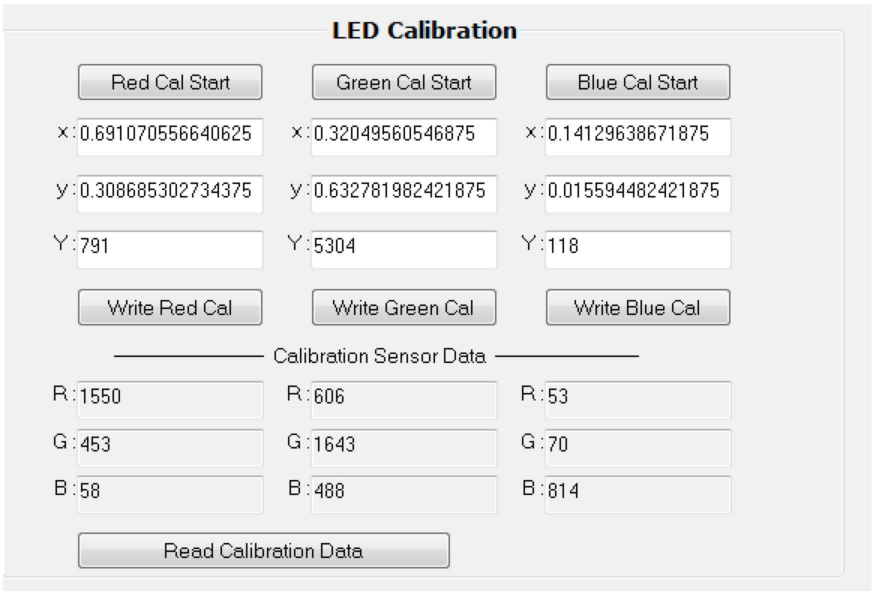
1. Select and run the brightest duty cycle from the available Looks. The result is the weakest color keep at maximum LED current level (such as 1023) for the given desired white point after running the brightest duty cycle.
2. Perform the frame-basis white point maintenance loops by adjusting LED current. The reference code does LED current adjustment every 5 frames.

4.2 Calibration Procedure

The new WPC provides a very simple calibration procedure in the development phase and manufacturing process. The algorithm only needs 9 parameters of the x, y color coordinates and Y, the brightness for each R, G, B LED. The user can do the calibration procedure in any Looks and any Vsync frequency.

Figure 14 is the LED calibration GUI. The calibration procedure follows:

1. Press the **Red Cal Start** button. MSP430 send commands to DLPC343x to turn on only the Red LED and display red curtain.
2. Fill out x, y, Y reading from color meter, and press **Write Red Cal**.
3. Press the **Green Cal Start** button. MSP430 send commands to DLPC343x to turn on only the Green LED and display green curtain.
4. Fill out x, y, Y reading from color meter, and press **Write Green Cal**.
5. Press the **Blue Cal Start** button, MSP430 send commands to DLPC343x to turn on only the Blue LED and display blue curtain.
6. Fill out x, y, Y reading from color meter, and press the **Write Blue Cal**.
7. The calibration data is saved in the MSP430 embedded flash automatically, and New WPC is ready to go.



x:	0.691070556640625	0.32049560546875	0.14129638671875
y:	0.308685302734375	0.632781982421875	0.015594482421875
Y:	791	5304	118
R:	1550	606	53
G:	453	1643	70
B:	58	488	814

Figure 14. LED Calibration GUI

5 Application Functions User's Guide

This section lists all of the application functions associated with the DLP Composer GUI and I^{2}C command format. Contact your regional technical support representative to get the source code if needed.

All New WPC commands start with 0x36 device address followed by a 2-byte WPC command address. The first byte of the command address is fixed with 0xFF for distinguishing from the common DLPC343x commands.

5.1 *Write CAIC LUT Build (0xE0)*

Start building CAIC LED current versus intensity LUT for each R, G, B LED. The process may take a few seconds to a few minutes depending on the software setting of current steps and delay.

Table 1. 0xE0

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11100000 (0xE0) 0000000en	en: 1 = Start building CAIC LED current versus intensity LUT

Table 2. 0xE0 Function Call

void WPC_WriteCAICBuildStart(CAIC_LUTStatus Status)	
Parameters	Status: Idle, InProcess, or Complete

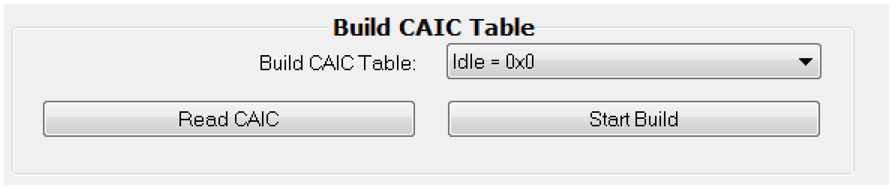


Figure 15. Build CAIC LUT GUI

5.2 *Read CAIC LUT Build Status (0xE1)*

Read the status of CAIC LUT build status.

Table 3. 0xE1

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11100001 (0xE1)	
Read	000000s1s0	s1 and s0: 00 = Idle 01 = In process 10 = Complete

Table 4. 0xE1 Function Call

CAIC_LUTStatus WPC_CAICBuildStatus(void)	
Parameters	Void

5.3 Read CAIC LUT Data (0xE2)

Read CAIC LED current versus intensity LUT. The LUT was stored with 4 data columns, LED current, red intensity, green intensity, and blue intensity. The LUT was read by frond-end and saved as a .tsv file before rebuilding the DLPC343x flash for download.

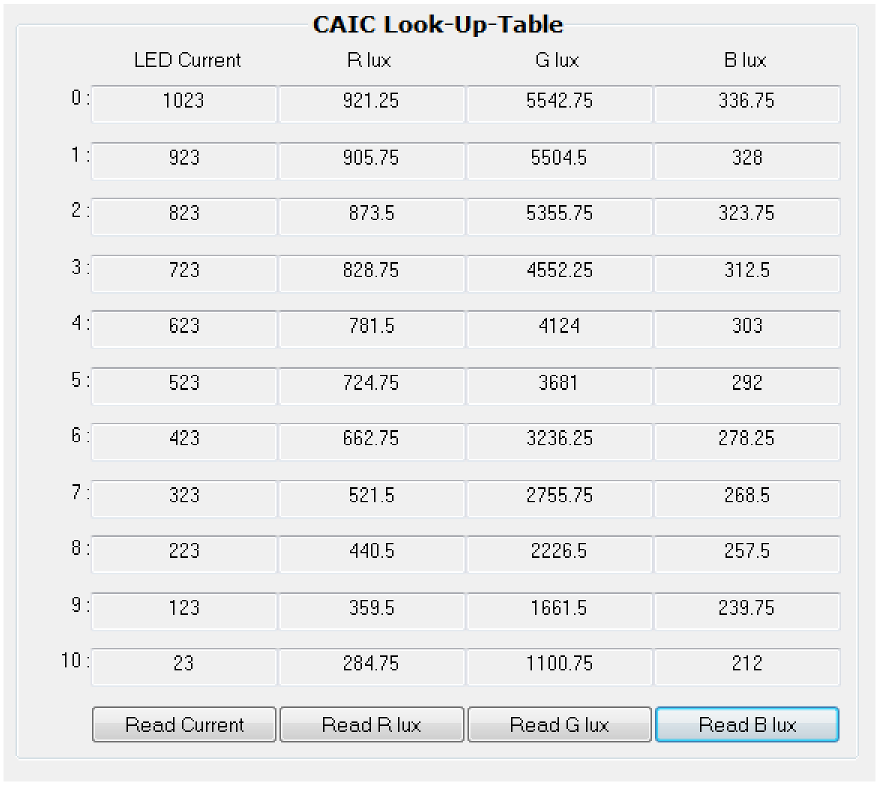
Table 5. 0xE2[(1)]

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11100010 (0xE2) 000000d1d0	d1 and d0: 00 = LED current column 01 = Red intensity 10 = Green intensity 11 = Blue intensity
Read	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ... _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	(0) data0' LSB (1) data0' MSB ... (n – 1) data(n / 2)' LSB (n) data(n / 2)' MSB

(1) data 0 = step 0's LED current or intensity
data n = step n's LED current or intensity

Table 6. 0xE2 Function Call

void WPC_ReadCAICTableData(char DataSelection)	
Parameters	DataSelection: LED current, red, green, or blue intensity


CAIC Look-Up-Table

	LED Current	R lux	G lux	B lux
0 :	1023	921.25	5542.75	336.75
1 :	923	905.75	5504.5	328
2 :	823	873.5	5355.75	323.75
3 :	723	828.75	4552.25	312.5
4 :	623	781.5	4124	303
5 :	523	724.75	3681	292
6 :	423	662.75	3236.25	278.25
7 :	323	521.5	2755.75	268.5
8 :	223	440.5	2226.5	257.5
9 :	123	359.5	1661.5	239.75
10 :	23	284.75	1100.75	212

Figure 16. CAIC LED Current versus Intensity LUT GUI (Read-Only)

5.4 Write Desired Color Coordinate (0xE8)

Set user’s desired color coordinates and gain, and calculate the appropriate P7 registers. CCA (0x86) should be enabled to validate the parameters. Each target color can be controlled independently. Set gain to 0 individually turn off the specific target color.

Table 7. 0xE8[1]

BYTE	COMMAND	NOTES

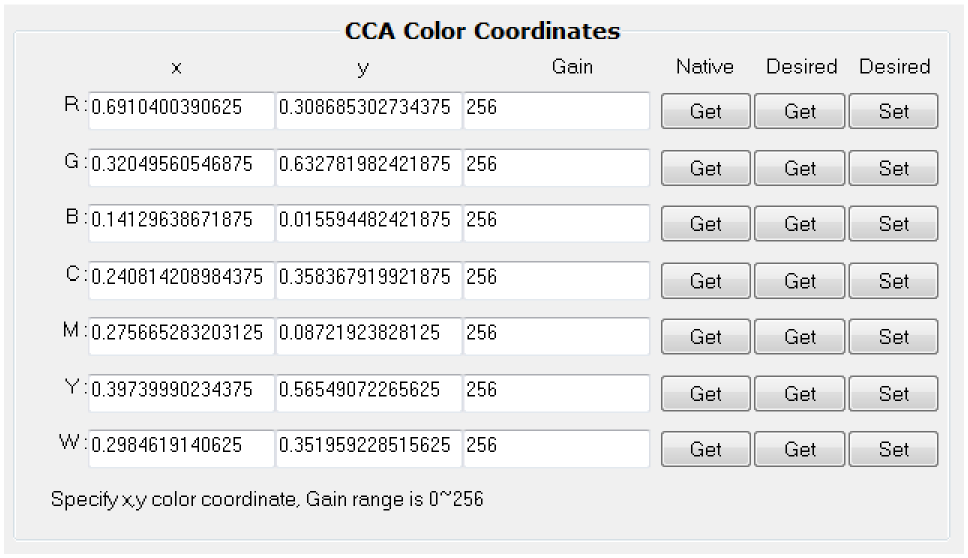
0:2	11111111 (0xFF)
	11101000 (0xE8)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

3:5	_ _	x-coordinate's LSB x-coordinate's MSB y-coordinate's LSB
6:8	_ _	y-coordinate's LSB clr's gain LSB clr's gain MSB

[1] Clr's x-coordinate: 1.15 fraction; Clr's y-coordinate: 1.15 fraction
Clr's gain: Gain value range 0 to 256

Table 8. 0xE8 Function Call

void WPC_WriteCCA_DesiredColor(PCC_COLOR Color, unsigned char* Value)	
Parameters	Color: the target color for adjustment *value: point of data structure of desired color coordinates and gain


CCA Color Coordinates

	x	y	Gain	Native	Desired	Desired
R:	0.6910400390625	0.308685302734375	256	Get	Get	Set
G:	0.32049560546875	0.632781982421875	256	Get	Get	Set
B:	0.14129638671875	0.015594482421875	256	Get	Get	Set
C:	0.240814208984375	0.358367919921875	256	Get	Get	Set
M:	0.275665283203125	0.08721923828125	256	Get	Get	Set
Y:	0.39739990234375	0.56549072265625	256	Get	Get	Set
W:	0.2984619140625	0.351959228515625	256	Get	Get	Set

Specify x,y color coordinate, Gain range is 0~256

Figure 17. CCA GUI

5.5 Read Desired Color Coordinate (0xE9)

The command read the target color's desired x,y coordinates, and gain which was previously set by 0xE8.

Table 9. 0xE9[1]

BYTE	COMMAND	NOTES

0:2	11111111 (0xFF)
	11101001 (0xE9)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

Read	_ _	x-coordinate's LSB x-coordinate's MSB y-coordinate's LSB y-coordinate's MSB clr's gain LSB clr's gain MSB

[1] Clr's x-coordinate: 1.15 fraction; Clr's y-coordinate: 1.15 fraction
Clr's gain: Gain value range 0 to 256

Table 10. 0xE9 Function Call

void WPC_ReadCCA_DesiredColor(PCC_COLOR Color)	
Parameters	Color: the target color to read the data

5.6 Read Native Color Coordinate (0xEA)

Read the native color coordinates for each color before CCA make up.

Table 11. 0xEA[1]

BYTE	COMMAND	NOTES

0:2	11111111 (0xFF)
	11101010 (0xEA)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

Read	_ _	x-coordinate's LSB x-coordinate's MSB y-coordinate's LSB y-coordinate's MSB clr's gain LSB clr's gain MSB

[1] Clr's x-coordinate: 1.15 fraction; Clr's y-coordinate: 1.15 fraction
Clr's gain: Gain value range 0 to 256

Table 12. 0xEA Function Call

void WPC_ReadCCA_NativeColor(PCC_COLOR Color)	
Parameters	Color: the target color to read the native color data

5.7 Write PCC CCA Register (0xEB)

Direct access P7 color elements, each color has red, green, and blue elements. The command allows the user to independently change the element to modify color appearance. CCA (0x86) should be enabled to validate the parameters.

Table 13. 0xEB[1]

BYTE	COMMAND	NOTES

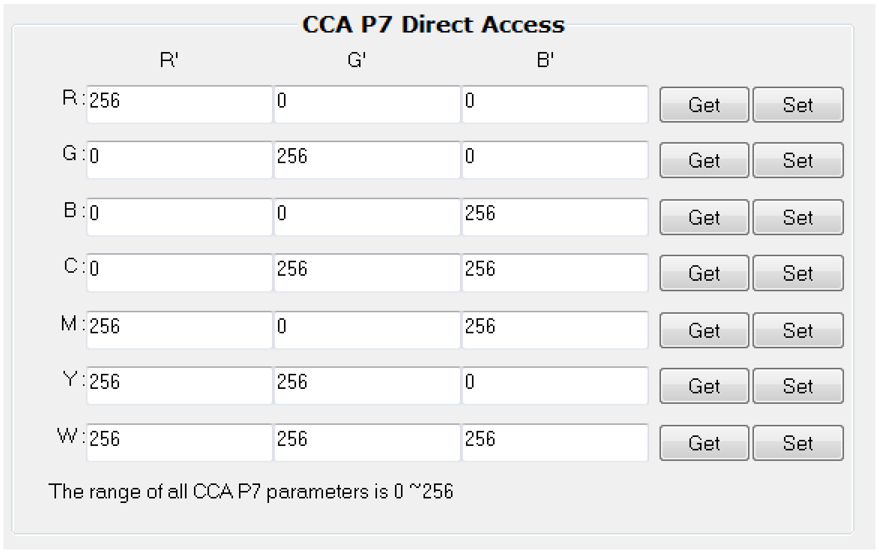
0:2	11111111 (0xFF)
	11101011 (0xEB)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

3:8	_ _	clr's R LSB clr's R MSB clr's G LSB clr's G LSB clr's B MSB clr's B LSB

[1] Clr's R: Red element (0 to 256)
Clr's G: Green element (0 to 256)
Clr's B: Blue element (0 to 256)

Table 14. 0xEB Function Call

void WPC_WritePCCRegister(PCC_COLOR Color, unsigned char* Value)	
Parameters	Color: the target color to read the native color data *Value: the point to access the value of red, green, and blue color element


CCA P7 Direct Access

	R'	G'	B'		
R :	256	0	0	Get	Set
G :	0	256	0	Get	Set
B :	0	0	256	Get	Set
C :	0	256	256	Get	Set
M :	256	0	256	Get	Set
Y :	256	256	0	Get	Set
W :	256	256	256	Get	Set

The range of all CCA P7 parameters is 0 ~256

Figure 18. CCA P7 Direct Register Access GUI

5.8 Read PCC CCA Register (0xEC)

The command read the target color's red, green, and blue element, which was previously set by 0xEB or Composer color adjustment settings.

Table 15. 0xEC[1]

BYTE	COMMAND	NOTES

0:2	11111111 (0xFF)
	11101100 (0xEC)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

Read	_ _	clr's R' LSB clr's R' MSB G' LSB G' MSB B' LSB B' MSB

[1] Clr's R': Red element (0 to 256)
G': Green element (0 to 256)
B': Blue element (0 to 256)

Table 16. 0xEC Function Call

void WPC_ReadPCCRegister(PCC_COLOR Color)	
Parameters	Color: the target color to read the P7 registers

5.9 Write Hue Saturation Gain (0xED)

The command allows the user to adjust each color by Hue(0° to 360°), Saturation(0 to 1), and Gain(0 to 256) before the HSG mechanism calculates the appropriate P7 registers. CCA (0x86) should be enabled to validate the parameters. Each target color can be controlled independently. Set gain to 0 individually to turn off the specific target color.

Table 17. 0xED[1]

BYTE	COMMAND	NOTES

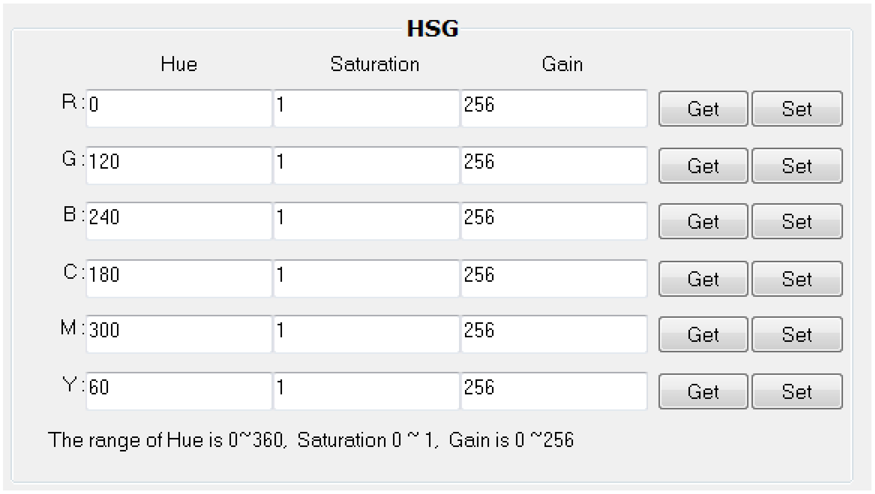
0:2	11111111 (0xFF)
	11101101 (0xED)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow;
		6 = White

3:5	_ _	clr's H LSB clr's H MSB clr's S LSB
6:8	_ _	clr's S MSB clr's G LSB clr's G MSB

[1] Clr's H: Hue adjustment (0° to 360°)
Clr's S: Saturation adjustment (0 to 1, 1.15 fraction)
Clr's G: Gain adjustment (0 to 256)

Table 18. 0xED Function Call

void WPC_WriteHSGRegister(PCC_COLOR Color, unsigned char* Value)	
Parameters	Color: the target color to read the native color data *Value: the point to access the value of Hue, Saturation, and Gain



	Hue	Saturation	Gain
R :	0	1	256
G :	120	1	256
B :	240	1	256
C :	180	1	256
M :	300	1	256
Y :	60	1	256

The range of Hue is 0~360, Saturation 0 ~ 1, Gain is 0 ~256

Figure 19. Hue Saturation Gain (HSG) GUI

5.10 Read Hue Saturation Gain (0xEE)

Table 19. 0xEE[1]

BYTE	COMMAND	NOTES

0:2	11111111 (0xFF)
	11101110 (0xEE)
	0000000clr
		Clr:
		0 = Red;
		1 = Green;
		2 = Blue;
		3 = Cyan;
		4 = Magenta;
		5 = Yellow

Read	_ _	clr's H' LSB clr's H' MSB clr's S' LSB clr's S' LSB clr's G' MSB clr's G' LSB

[1] Clr's H: Hue adjustment (0° to 360°)
Clr's S: Saturation adjustment (0 to 1, 1.15 fraction)
Clr's G: Gain adjustment (0 to 256)

Table 20. 0xEE Function Call

void WPC_ReadHSGRegister(PCC_COLOR Color)	
Parameters	Color: the target color to read Hue, Saturation, and Gain values

5.11 Write Calibration Start (0xF0)

The command starts the LED and color sensor calibration process. DLPC343x is set to display an internal solid field pattern with maximum LED current. CCA and WPC are disabled accordingly. Calibrate one LED at a time.

Table 21. 0xF0[(1)]

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11110000 (0xF0) 0000000clr	Clr 0 = Red 1 = Green 2 = Blue

(1)

Table 22. 0xF0 Function Call

void WPC_WriteCalibrationStart(LED_ColorType color)	
Parameters	Color: the target LED to calibrate. Red, Green, or Blue LED

5.12 Write Calibration Data (0xF1)

Write x,y,Y reading from the color meter; the calibration data is saved into MSP430 embedded flash. DLPC343x turns all LEDs back on after the process is complete. Set Write Calibration Start (0xF0) prior to 0xF1.

Table 23. 0xF1[(1)]

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11110001 (0xF1) 0000000clr	Clr 0 = Red 1 = Green 2 = Blue
3:5	_ _	clr's x LSB clr's x MSB clr's y LSB
6:8	_ _	clr's y MSB clr's Y LSB clr's Y MSB

(1) Clr's x: x-coordinate 1.15 fraction
Clr's y: y-coordinate 1.15 fraction
Clr's Y: Y stimulus 0 to 65535

Table 24. 0xF1 Function Call

void WPC_WriteLEDCalData	(LED_ColorType color, unsigned char *RGB_xyY_cal_data, uint16_t *Sensor_Counter_data, unsigned int FrameTime, BOOL FactoryCal)
Parameters	Color: the target LED to calibrate. Red, Green, or Blue LED *RGB_xyY_cal_data: the point of x,y,Y calibration data from the color meter *Sensor_Counter_data: the point of RGB color sensor reading FrameTime: the time period of two vsync rising edges FactoryCal: write the data from calibration process or power on initialization

5.13 Write Desired White Point (0xF3)

The command sets the desired white point color coordinates. The frame-basis WPC (0xFA) should be enabled to validate the command.

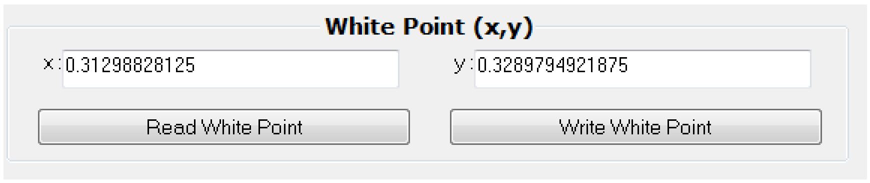
Table 25. 0xF3[1]

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11110011 (0xF3) _ _ _ _ _ _ _ _	x-coordinate LSB
3:5	_ _	x-coordinate's MSB y-coordinate's LSB y-coordinate's MSB

[1] x-coordinate: 1.15 fraction
y-coordinate: 1.15 fraction

Table 26. 0xF3 Function Call

void WPC_WriteWhitePoint(unsigned char* Target_xy)	
Parameters	*Target_xy: the point of the value of desired white point



x : 0.31298828125
y : 0.3289794921875

Figure 20. Desired White Point GUI

5.14 Read Desired White Point (0xF4)

The command reads the desired white point which was previously set by White Desired White point (0xF3).

Table 27. 0xF4[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11110100 (0xF4)	
Read	_ _	(0) x-coordinate's LSB (1) x-coordinate's MSB (2) y-coordinate's LSB (3) y-coordinate's MSB

[1] x-coordinate: 1.15 fraction
y-coordinate: 1.15 fraction

5.15 Write Max LED Current (0xF5)

The command sets the max LED current when frame-basis WPC (0xFA) is enabled. The max value for each LED can be different, and frame-basis WPC manages the desired white point by adjusting LED current under or equal to the value.

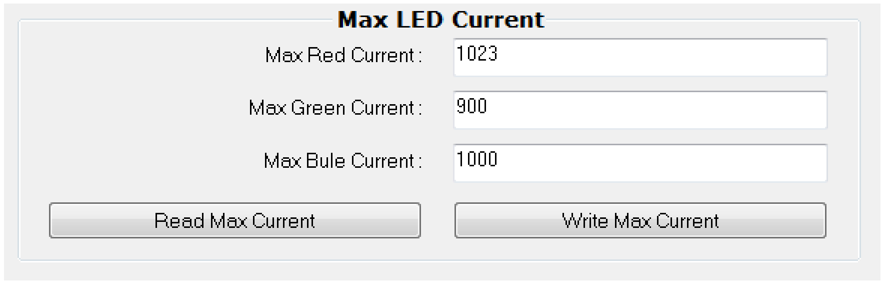
Table 28. 0xF5[1]

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11110101 (0xF5) _ _ _ _ _ _ _ _	 red's max LSB
3:5	_ _	red's max MSB green's max LSB green's max MSB
6:8	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	blue's max LSB blue's max MSB

[1] red's max: Max red LED current (0 to 1023);
green's max: Max green LED current (0 to 1023);
blue's max: Max blue LED current (0 to 1023)

Table 29. 0xF5 Function Call

void WPC_WriteMaximumLEDcurrent(unsigned char* MaxLEDcurrent)	
Parameters	* MaxLEDcurrent _xy: the point of the value of max LED current



Max Red Current : 1023
Max Green Current : 900
Max Bule Current : 1000

Figure 21. Max LED Current GUI

5.16 Read Max LED Current (0xF6)

The command reads the max LED current, which was previously set by White Max LED current (0xF5).

Table 30. 0xF6[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11110110 (0xF6)	
Read	_ _	(0) red's max LSB (1) red's max MSB (2) green's max LSB (3) green's max MSB (4) blue's max LSB (5) blue's max MSB

[1] red's max: Max red LED current (0 to 1023);
green's max: Max green LED current (0 to 1023);
blue's max: Max blue LED current (0 to 1023)

Table 31. 0xF6 Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	Void

5.17 Read Sensor Pulse Count (0xF7)

The command reads the present RGB sensor pulse counts. The integration time is 5-frame time as default. TI suggests that sensor pulse counts reach at least 500.

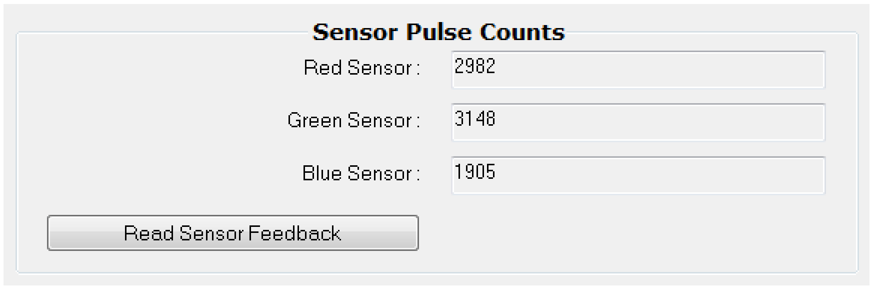
Table 32. 0xF7[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11110111 (0xF7)	
Read	_ _	(0) red S's LSB (1) red S's MSB (2) green S's LSB (3) green S's LSB (4) blue S's MSB (5) blue S's LSB

[1] red S's: Red sensor pulse counts
green S's: Green sensor pulse counts
blue S's: Blue sensor pulse counts

Table 33. 0xF7 Function Call

void WPC_ReadSensorPulseCount(void)	
Parameters	void


Sensor Pulse Counts

Red Sensor: 2982

Green Sensor: 3148

Blue Sensor: 1905

Figure 22. Real-Time Sensor Pulse Count GUI (Read-Only)

5.18 Read x,y,Y and Color Temp (0xF8)

The command reads the present x,y color coordinates, Y stimulus, and color temperature (K) data from LED illuminator. LED and color sensor (0xF1) calibration need to be complete prior to using the command.

Table 34. 0xF8[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11111000 (0xF8)	
Read	– –	(0) x-coordinate's LSB (1) x-coordinate's MSB (2) y-coordinate's LSB (3) y-coordinate's MSB (4) Y stimulus's LSB (5) Y stimulus's MSB (6) Clr temp's LSB (7) Clr temp's MSB

[1] x-coordinate: 1.15 fraction
y-coordinate: 1.15 fraction
Y stimulus
Clr temp: color temperature (K)

Table 35. 0xF8 Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void

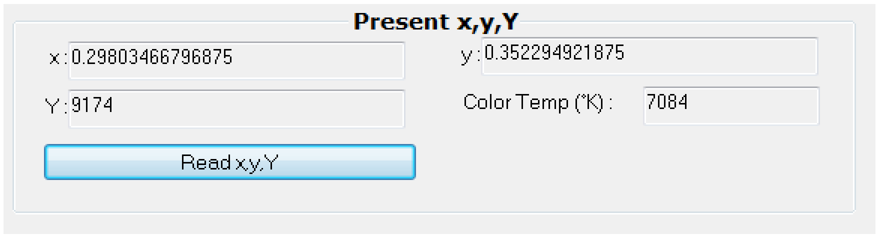


Figure 23. Real-Time x, y, Y and Color Temp Reading GUI

5.19 Read LED Efficiency (0xF9)

The command allows the user to monitor each LED emission efficiency (in %). The comparison base (100%) is the moment that saves LED calibration data.

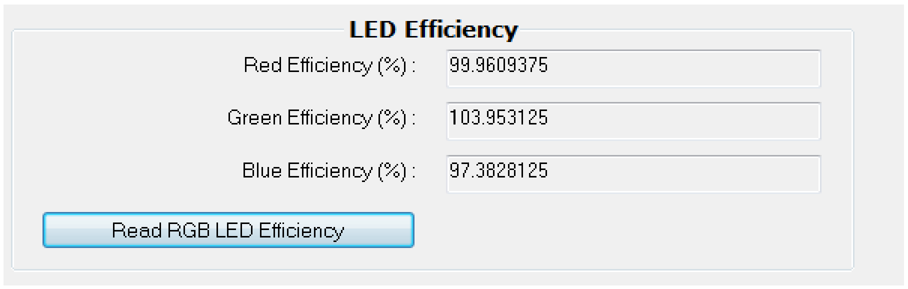
Table 36. 0xF9[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11110001 (0xF9)	
Read	_ _	(0) red's eff LSB (1) red's eff MSB (2) green's eff LSB (3) green's eff MSB (4) blue's eff LSB (5) blue's eff MSB

[1] red's eff: Red LED efficiency (8.8 fraction)
green's eff: Green LED efficiency (8.8 fraction)
blue's eff: Blue LED efficiency (8.8 fraction)

Table 37. 0xF9 Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void



Red Efficiency (%) : 99.9609375
Green Efficiency (%) : 103.953125
Blue Efficiency (%) : 97.3828125

Figure 24. Real-Time LED Efficiency GUI

5.20 Write Real-Time WPC Enable (0xFA)

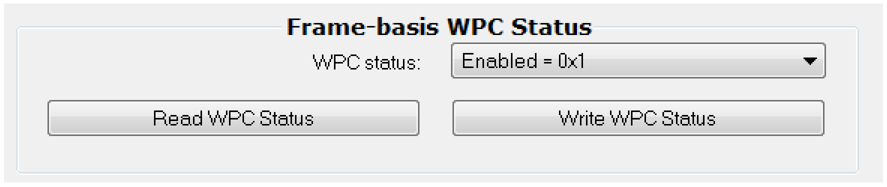
Enable real-time white point correction by adjusting LED current every 5 frames (default). The desired white point is set by 0xF4 and manages LED current under the setting of max LED current 0xF5. Disable CAIC if WPC is enabled.

Table 38. 0xFA

BYTE	COMMAND	NOTES
0:2	11111111 (0xFF) 11111010 (0xFA) 0000000en	en: 0 = Disable WPC; 1 = Enable WPC

Table 39. 0xFA Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void



WPC status: Enabled = 0x1

Figure 25. Real-Time WPC Setting GUI

5.21 Read Real-Time WPC Enable (0xFB)

Read the status of real-time WPC.

Table 40. 0xFB

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11111011 (0xFB)	
Read	0000000en	en: 0 = Disable WPC; 1 = Enable WPC

Table 41. 0xFB Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void

5.22 Read WPC Library Version (0xFC)

Read the encrypted WPC library verison; The WPC library is managed by TI.

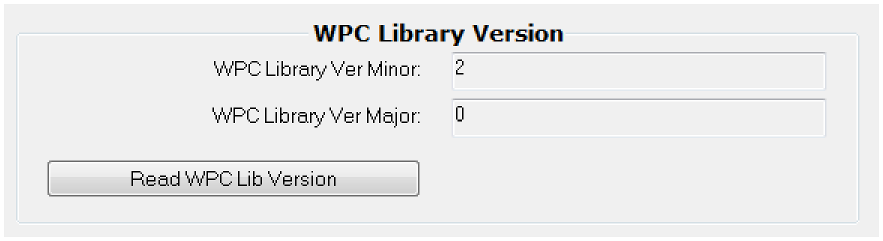
Table 42. 0xFC[1]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11111100 (0xFC)	
Read	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	(0) Lib ver minor (1) Lib ver major

(1)

Table 43. 0xFC Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void



WPC Library Ver Minor: 2

WPC Library Ver Major: 0

Figure 26. Encrypted WPC Library Version

5.23 Read WPC Application Version (0xFD)

Read the WPC application version; the version is changeable in WPC source code.

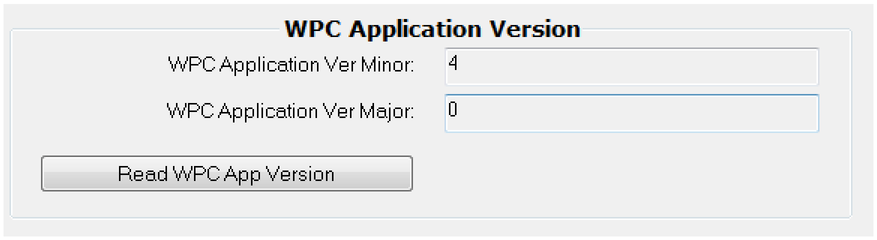
Table 44. 0xFD[(1)]

BYTE	COMMAND	NOTES
0:1	11111111 (0xFF) 11111101 (0xFD)	
Read	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	(0) WPC app ver minor (1) WPC app ver major

(1)

Table 45. 0xFD Function Call

void WPC_ReadMaximumLEDcurrent(void)	
Parameters	void


WPC Application Version

WPC Application Ver Minor: 4

WPC Application Ver Major: 0

Figure 27. WPC Application Version

Table 46. New WPC Commands Quick Reference Table

COMMAND	R/W	NOTES
0xFF 0xE0	W	Write CAIC LUT build
0xFF 0xE1	R	Read CAIC LUT build status
0xFF 0xE2	R	Read CAIC LUT data
0xFF 0xE8	W	Write desired color coordinate
0xFF 0xE9	R	Read desired color coordinate
0xFF 0xEA	R	Read native color coordinate
0xFF 0xEB	W	Write PCC CCA register
0xFF 0xEC	R	Read PCC CCA register
0xFF 0xED	R	Write hue saturation gain
0xFF 0xEE	R	Read hue saturation gain
0xFF 0xF0	W	Write calibration start
0xFF 0xF1	W	Write calibration data
0xFF 0xF2	R	Read calibration data
0xFF 0xF3	W	Write desired white point
0xFF 0xF4	R	Read desired white point
0xFF 0xF5	W	Write max LED current
0xFF 0xF6	R	Read max LED current
0xFF 0xF7	R	Read sensor pulse count
0xFF 0xF8	R	Read x, y, Y and color temperature
0xFF 0xF9	R	Read LED efficiency
0xFF 0xFA	W	Write real-time WPC enable
0xFF 0xFB	R	Read real-time WPC enable
0xFF 0xFC	R	Read WPC library version
0xFF 0xFD	R	Read WPC application version

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		