Application Report
SPRABR7–February 2013

Multi-DC/DC Conversion and Color LED Control Integrated on a C2000™ Microcontroller

Daniel Chang

ABSTRACT

This application report presents a solution to control an LED lighting system using TMS320F2802x microcontrollers. The Piccolo™ TMS320F2802x series of devices are part of the family of C2000 microcontrollers, which enables cost-effective designs of LED lighting systems. With these devices, it is possible to control multiple strings of LEDs in an efficient and very accurate way. In addition to this, the speed of the C2000 microcontroller allows it to integrate many supplemental tasks that would, in a normal system, increase chip count and complexity. These tasks could include DC/DC conversion stages, AC/DC conversion stages with PFC, system management, and various communication protocols such as DALI, DMX512, KNX or even power line communication (PLC). On the board described in this document, eight independent DC/DC stages (6x Boost, 2x Sepic) are use to drive eight individual LED strings.

This application report covers the following:

- A brief overview of LED Lighting technology
- The advantages C2000 can bring to a lighting system
- How to run and get familiar with the Multi-DC/DC Color LED Kit's LED-ColorMix project

Contents

List of Figures

List of Tables

1 LED Lighting Theory

1.1 The Benefit of LEDs

As a relatively new and developing technology, LEDs have already become a valid solution in many lighting applications. One major advantage LEDs bring to applications is their high efficiency. Today's high brightness LEDs have a luminous efficiency of up to 60 lm/W with claims of greater than 100 lm/W being made from various LED manufacturers. Another advantage LEDs have over other light sources is their extensive life, on the order of 50,000 hours if designed correctly. Since, in applications such as street lighting bulb replacement can be quite expensive when labor and loss of service are considered, LEDs can be seen as having an advantage in these spaces. LEDs also have excellent vibration resilience, provide directional lighting, and allow for almost full dimming ability. These features, and more, allow LEDs to be perceived as the future in lighting technology.

1.2 Light and LED Characteristics

All light has two major characteristics: luminous flux and chromaticity. Luminous flux is an attribute of visual perception in which a source appears to be radiating or reflecting light. The term "brightness" is often given to describe this characteristic, however, this term is often used in a physiological and non-quantitative way. A better term is luminous flux which, measured in lumens, is the light power measured multiplied with the V-λ scaling function. This function is used to compensate for the human eye's sensitivity to different wavelengths.

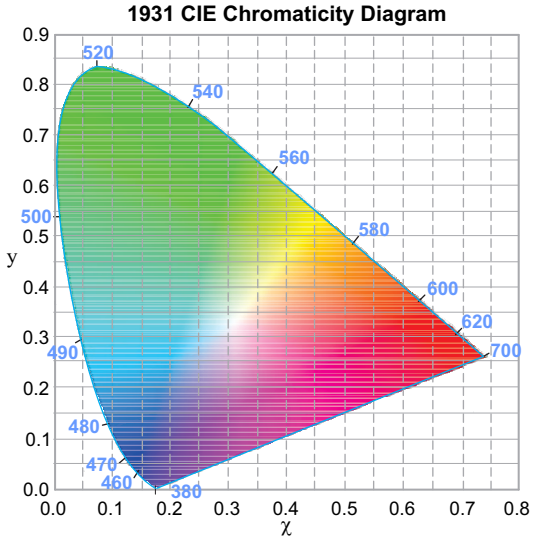


Figure 1. 1931 CIE Chromaticity Diagram

LEDs can be seen as a current-controlled device. The luminous flux and chromaticity are largely governed by the current that flows through the device. In Figure 2, it can be seen that current is nearly proportional to the luminous flux output from an LED. Because of this fact, in many systems the average current is edited in order to dim an LED or LED string. The only other major variable that can affect luminous flux and chromaticity is temperature. Because of this, it is important to control temperature changes in an LED system.

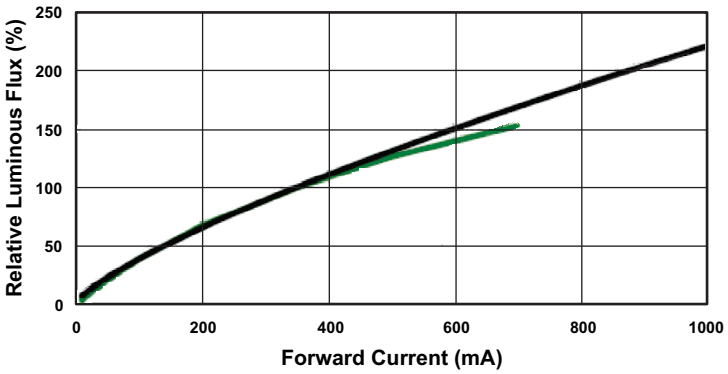


Figure 2. LED Current vs. Luminous Flux

Figure 3 shows the relationship between forward voltage and current in an LED. Note that the LED behaves similarly to a diode in that it requires a certain threshold voltage before it will begin conducting. Once the forward voltage becomes greater than the threshold voltage the current will increase exponentially until it reaches the maximum current specification of the LED device. For a string of six LEDs, it would be necessary to make the forward voltage larger than eight times the LED's threshold voltage in order to make the LEDs light up.

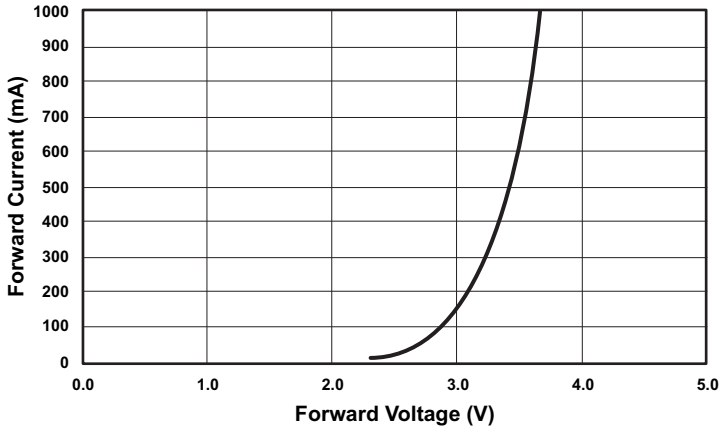


Figure 3. LED Forward Voltage vs. Current

Because of the LED's relationship between voltage and current it may seem reasonable that one could control the LED voltage in order to specify the LED current and, therefore, the luminous flux given by the LED string. The problem here is that as temperature even slightly increases the graph above will keep the same shape, but shift to the left. For instance, if one LED (with the specifications of Figure 3) was controlled at 3.0 V about 150 mA of current would be drawn through the LED initially. However, as the LED warmed up, the graph will shift to the left and the current would increase slightly. This increased current would then increase the temperature. This cycle would continue until the LED device failed. Because of this, current control of LED strings is highly preferred. It also helps to show the importance of thermal management in an LED system.

1.3 Control Techniques

There are various techniques for controlling the current for an LED string. In many cases a simple solution may be adequate, but for many cases the number of LEDs in a string may be large or the total luminous flux needed is expected to be large. In order to maintain efficiency, a switched-mode power supply will likely be needed. Figure 4 shows one method of controlling one of these systems.

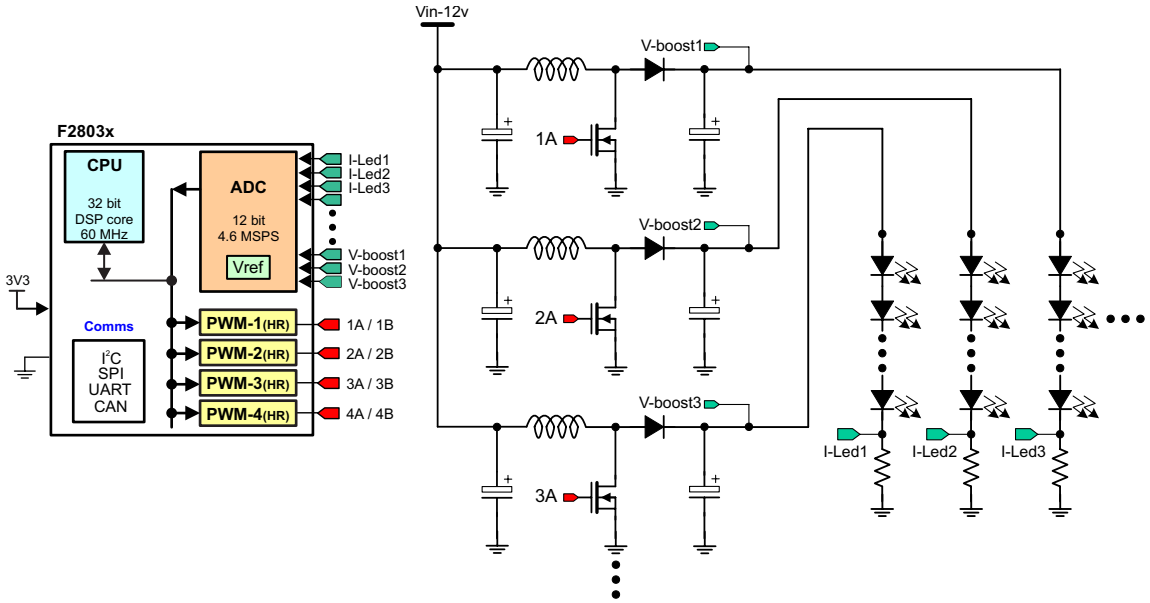


Figure 4. Individual Power Stage per String

In the solution above, each string is powered by a current-controlled boost stage. The Piccolo device provides the feedback loop by sampling each string's current from a resistor placed in series with the LED string. This sampled current is then compared to a reference within the controller and this helps to determine the next value of duty cycle sent to the MOSFETs in each individual DC/DC boost stage. In order to maintain the constant current needed, the pulse width modulation (PWM) frequency of the FETs must be relatively large in order to prevent flickering.

The high-performance peripherals available on a Piccolo device can control the system above and still have a large amount of bandwidth to provide system management. The Piccolo peripherals, namely the on-chip comparator and the configurability of ADC sampling, allow the MCU to control the current via peak or average current mode, respectively.

On the following page, a different technique is shown. A single DC/DC stage is used to provide the voltage that will appear across the LED strings. This voltage will then correspond to a single current that will pass through an LED string. Each individual string is then switched on and off by a MOSFET so that the average power sent through an LED string is decreased. In this way, each individual LED string can be dimmed to a reference average current.

On the DC/DC LED lighting kit, this latter approach has been implemented. The single DC/DC stage is a SEPIC converter and the Piccolo MCU also controls up to eight LED strings.

In either of these strategies, string interleaving can be done to reduce the peak current and improve the overall efficiency. Piccolo's PWMs allow for synchronization between individual PWM modules and/or with an external synchronization pulse.

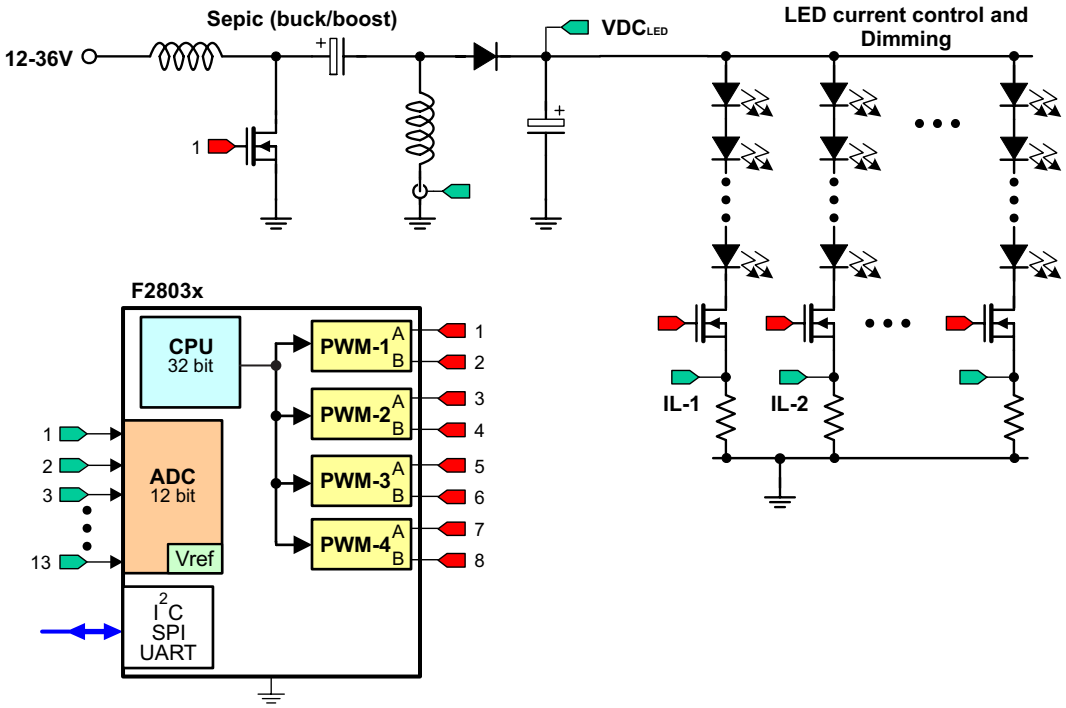


Figure 5. PWM Dimmed Strings With a Common Supply

1.4 An Efficient LED System

Any LED system must have power conversion stages to deliver the correct voltage and current to the LED strings. A typical system, like the one shown in Figure 6, will have an AC/DC rectifier followed by a PFC boost circuit, and then one or more parallel DC/DC stages will be used to drive one or more LED strings. To create an efficient system, each of these power stages must be designed to be efficient and the control of these power stages must be efficient.

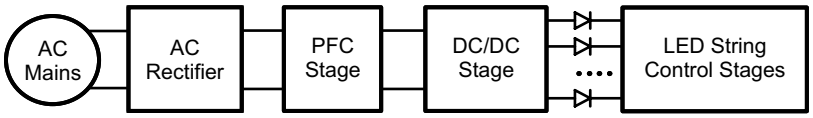


Figure 6. Lighting System

Communications also play a large role in the development of an intelligent, efficient system. A central area, or an application based on a mobile phone could control the lighting in a home or office without the need to string control them individually. Many lighting standards already exist such as DALI, KNX, and DMX512. These, as well as others, exist as a twisted pair, wireless, or power-line-communication (PLC) solutions.

With the advent of new and cheap MCUs like the C2000's Piccolo series and MSP430, an efficient system can be made just by improving power stages and by using the most advanced LEDs. Intelligent lighting, in which the system is aware of its surroundings or is controlled by a networked host, can also play a large role in improving efficiency and reducing maintenance costs. Individual ballasts could use light sensors to sense ambient light and only generate the amount of light the area needs. An MCU could count the lifetime of an LED and compensate for the degeneration of light output with time and warn an external system if it has reached a pre-determined lifetime, in which the ballast could begin showing signs of failure.

With the high performance and software flexibility of the C2000 Piccolo series, string control, power stage control, communication, and intelligent lighting control can all be done on a single chip. Obviously, processor bandwidth limits what can be done to some point, but in many systems the LED string controls, DC/DC stage, and some system control uses less than 50% of the CPU bandwidth on the C2000.

2 Benefits of C2000 in LED Lighting

The C2000 family of devices possess the desired computation power to execute complex control algorithms and the right mix of peripherals needed in power stage control and LED string control. These peripherals have all the necessary hooks for implementing systems that meet safety requirements, like on-chip comparators and trip zones for the PWMs. Along with this, the C2000 ecosystem of software (libraries and application software) and hardware (application kits) help to reduce the time and effort needed to develop a lighting solution.

A C2000 lighting solution can provide the following benefits:

- Precise current matching between strings
- Accurate color matching
- Large range of dimming ability; greater than 20,000:1 is possible
- Can perform PWM or constant current dimming
- Amount of LEDs in a string is not limited by the device. If a greater number of LEDs is needed in a particular product, only the MOSFET and MOSFET drivers would need to be checked and the software changes should be minor.
- Adaptive dimming based on LED usage, aging, sensed external brightness, and so forth
- Easily synchronized to video clock via CAP and QEP peripherals
- Efficient control of PFC, AC/DC, DC/DC and more due to the power and flexibility of the C2000 peripheral set. Power supply control is a major strength of C2000 in the market.
- Large range of communication protocols such as inter-integrated circuit (I2C), serial peripheral interface (SPI) and universal asynchronous receiver/transmitter (UART)
- High performance CPU allows the C2000 devices to do multiple tasks such as individual string control of multiple strings, DC/DC control, AC/DC control and communications on one chip.
- Because the device is programmed via software, creating products for multiple regions and multiple configurations often requires only minor changes in software.

3 System Overview

3.1 Hardware Overview

The Multi-DC/DC Color LED Kit is designed to take in 12 V – 18 V DC (nominal) when used with the included LED panel. The output current for each of the eight DC/DC power stages is independently regulated to a desired level in order to achieve the brightness and color(s) desired from the LED strings since the brightness of an LED is roughly proportional to the current. For this application, the output currents are set in the range of 0-50 mA per string. To perform independent LED string control, each LED string is connected to a separate DC/DC power stage. Every DC/DC power stage contains a MOSFET and the on-time of this MOSFET controls the average current through an LED string. Figure 7 illustrates the hardware present on the Multi-DC/DC Color LED Kit.

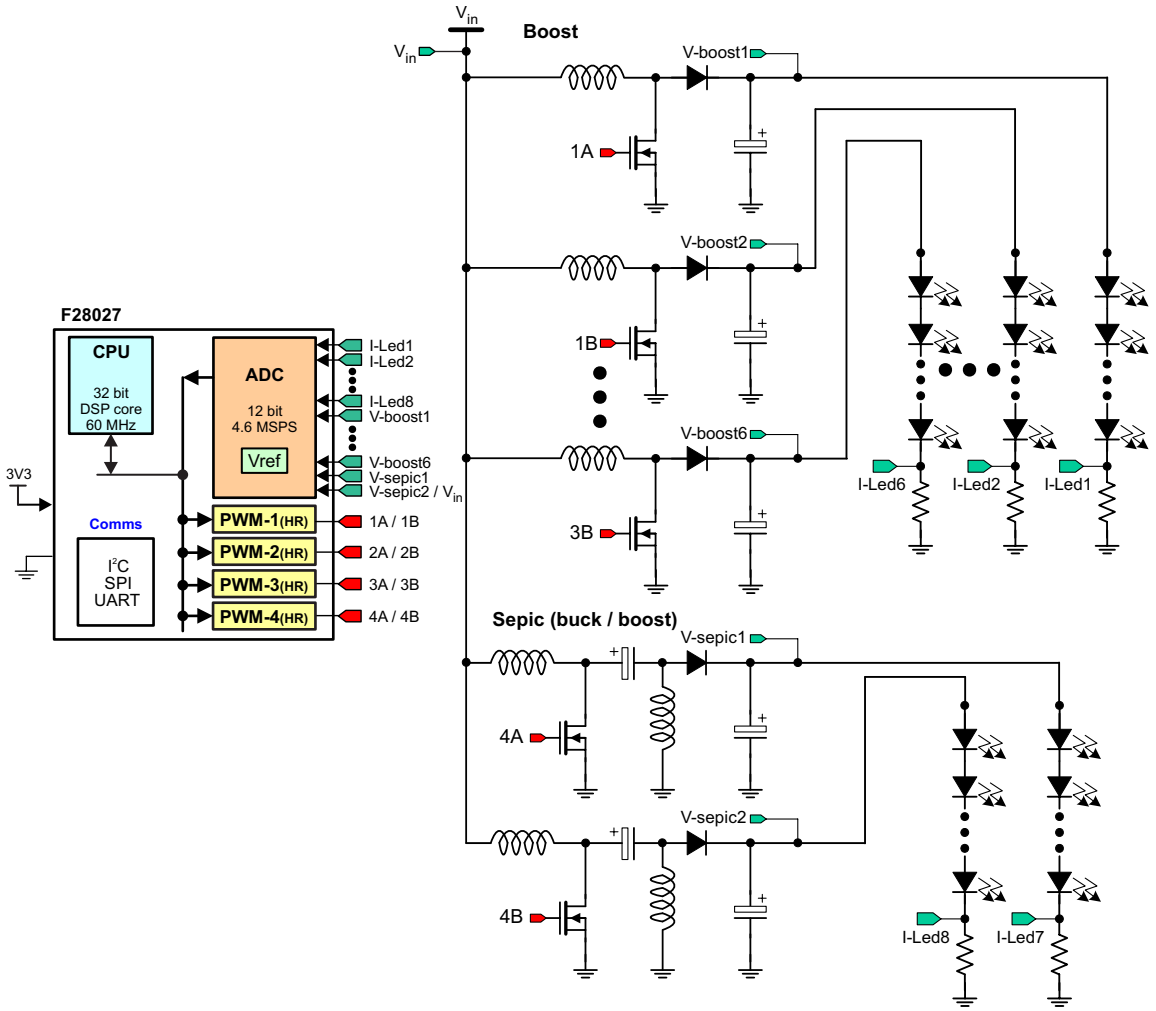


Figure 7. Multi-DC/DC Color LED Kit Circuit Diagram

The key signal connections between the F28027 microcontroller and the Multi-DC/DC Color LED Kit board are listed in Table 1.

Table 1. PWM and ADC Resource Allocation

Macro Name			Signal Name	PWM/ADC Channel	Description
Main Board	**[Main]**	**VDCin-meas**	**VDCin-meas**	**ADC-B4**	**Input Voltage Sense Muxed With V-Led-4B**
Dual Boost	[M2]	PWM-1	PWM-1A	PWM-1A	Boost 1 PWM signal
		PWM-2	PWM-1B	PWM-1B	Boost 2 PWM signal
		Vfb-1	V-Led-1A	ADC-B1	Boost 1 output voltage sense
		Vfb-2	V-Led-1B	ADC-B2	Boost 2 output voltage sense
		Ifb-1	I-Led-1A	ADC-A2	Boost 1 output current sense
		Ifb-2	I-Led-1B	ADC-A0	Boost 2 output current sense
Dual Boost	[M3]	PWM-1	PWM-2A	PWM-2A	Boost 3 PWM signal
		PWM-2	PWM-2B	PWM-2B	Boost 4 PWM signal
		Vfb-1	V-Led-2A	ADC-B3	Boost 3 output voltage sense
		Vfb-2	V-Led-2B		Boost 4 output voltage sense
		Ifb-1	I-Led-2A	ADC-A4	Boost 3 output current sense
		Ifb-2	I-Led-2B	ADC-A1	Boost 4 output current sense
Dual Boost	[M4]	PWM-1	PWM-3A	PWM-3A	Boost 5 PWM signal
		PWM-2	PWM-3B	PWM-3B	Boost 6 PWM signal
		Vfb-1	V-Led-3A	ADC-B5	Boost 5 output voltage sense
		Vfb-2	V-Led-3B		Boost 6 output voltage sense
		Ifb-1	I-Led-3A	ADC-A6	Boost 5 output current sense
		Ifb-2	I-Led-3B	ADC-A3	Boost 6 output current sense
Dual Sepic	[M5]	PWM-1	PWM-4A	PWM-4A	Sepic 1 PWM signal
		PWM-2	PWM-4B	PWM-4B	Sepic 2 PWM signal
		Vfb-1	V-Led-4A	ADC-B7	Sepic 1 output voltage sense
		Vfb-2	V-Led-4B	ADC-B4	Sepic 2 output voltage sense muxed with VDCin-meas
		Ifb-1	I-Led-4A	ADC-B6	Sepic 1 output current sense
		Ifb-2	I-Led-4B	ADC-A7	Sepic 2 output current sense

4 Software Overview

4.1 Build Options

In order for you to slowly build up and understand the project, the project is divided into various builds seperated by #if options in the LED-ColorMix-Main.c and LED-ColorMix-ISR.asm files. Which build is used is set by the variable INCR_BUILD in LED-ColorMix-Settings.h. Below is a short description of the different builds available in the LED-ColorMix project.

- Build 1: Open Loop Test, Check basic operation of the DC/DC stages and LED strings
- Build 2: Closed Loop operation of the DC/DC stages and LED strings

Table 2. Major Variables

Gui_Vin	The DC input voltage (0 V – 36 V), for Instrumentation only
Gui_Vout{X}	The output voltage (0 V – 50 V) for DC/DC stage {X}, for Instrumentation only
Gui_Iout{X}	Each LED string's output current (0 A – 0.650 A), for Instrumentation only
Gui_Iset{X} (Build 2 only)	Sets the closed loop current reference (0-0.100A) for LED string {X}

Table 2. Major Variables (continued)

Pgain{X} (Build 2 only)	Proportional gain for LED string {X}; value adjustment : 0 ~ 1000
Igain{X} (Build 2 only)	Integral gain for LED string {X}; value adjustment : 0 ~ 1000
Dgain{X} (Build 2 only)	Derivative gain for LED string {X}; value adjustment : 0 ~ 1000
SlewStep{X} (Build 2 only)	This value determines how fast the LED current {X} will ramp to the reference current
SlewStepAll (Build 2 only)	This value determines how fast all LED currents will ramp to the reference current when changed
Duty{X} (Build 1 only)	Sets the open loop duty cycle of the DC/DC stage for LED string {X}
ChannelEnable{X} (Build 2 only)	Enables the output for LED string {X}
EnableAll (Build 2 only)	Enables the output of all LED strings
StopAll (Build 2 only)	Disables the output of all LED strings
MergeLEDs (Build 2 only)	• 0: LEDs are individually controlled by Gui_Iset{X} • 1: LEDs are controlled in pairs; 1 and 4 by Gui_Iset1, 2 and 5 by Gui_Iset2, 3 and 6 by Gui_Iset3, 7 and 8 by Gui_Iset7 (paired by color)
OCtrip (Build 2 only)	Indicates if the slow over-current trip was triggered and disables the output of all LED strings
OC_limit (Build 2 only)	This value determines the minimum output current needed to trip the slow over-current protection

4.2 Key Files Located in the Project

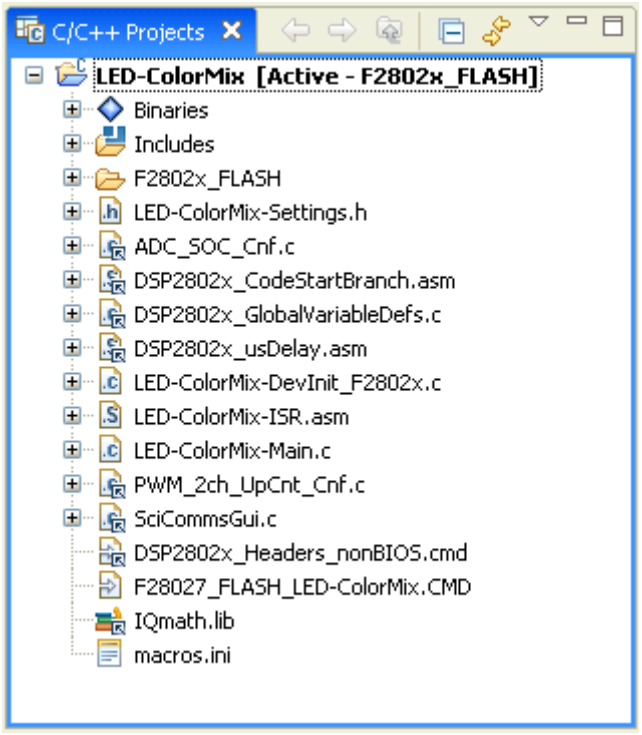


Figure 8. LED-ColorMix Project Files

The key framework files used in this project are:

- LED-ColorMix-Main.c – this file is used to initialize, run, and manage the application. This is the "brains" behind the application.
- LED-ColorMix-DevInit_F2802x.c – this file is responsible for initialization and configuration of the device (in this case the F28027), and includes functions for setting up the clocks, PLL, GPIO, and so forth.
- LED-ColorMix-ISR.asm – this file contains timing critical "control type" code. This file has an initialization section that is executed one time by the C-callable assembly subroutine _DPL_Init. The file also contains the _DPL_ISR routine which executes at a rate determined by the PWM or timer used to trigger it.
- LED-ColorMix-Settings.h – this file is used to set global definitions for the project (ie. build options). Note that it is linked to both LED-ColorMix-Main.c and LED-ColorMix-ISR.asm.
- LED-ColorMix-Calculations.xls – this is a spreadsheet file that calculates the values of the various scaling factors used in converting Q-value numbers, used by the MCU, to real world values. The variables K_Iout and iK_Iset are examples of scaling factors.

4.3 Macros Used

To reduce the effort for developing code each time, an approach of using Macro Blocks is used. These macro blocks are written in C-callable assembly and can have a C and assembly component. Table 3 provides a list of macros being used in this project.

Table 3. Macros Used

C Configuration Function	ASM Initialization Macro	ASM Run Time Macro
PWM_2ch_UpCnt_Cnf	PWMDRV_2ch_UpCnt_INIT n	PWMDRV_2ch_UpCnt n
None	CNTL_2P2Z_INIT n	CNTL_2P2Z n
ADC_SOC_Cnf	ADCDRV_1ch_INIT n	ADCDRV_1ch n

As the configuration of the peripherals is done in C, it can be easily modified. The ASM driver macro provides the necessary compact code to run in real time. Below is a list of each of the macros to better understand its role in the project.

- **PWM_2ch_UpCnt_Cnf()**

  Defined in PWM_2ch_UpCnt_Cnf.c, this function configures the PWM channels as specified in the arguments, to drive the power stage. For details on how and what arguments are passed, see the source file. This is a macro that is part of the TI PowerLib.

- **PWMDRV_2ch_UpCnt n**

  Defined in PWMDRV_2ch_UpCnt.asm, this macro is used to update PWM[n]A and PWM[n]B in the ISR as configured by the CNF file. This is a macro that is part of the TI PowerLib.

- **CNTL_2P2Z n**

  Defined in CNTL_2P2Z.asm, this macro is a second order compensator realized from an IIR filter structure. The five coefficients needed for this function are declared in the C background loop as an array of longs. This function is independent of any peripherals and, therefore, does not require a CNF function call. This is a macro that is part of the TI PowerLib.

- **ADC_SOC_Cnf()**

  Defined in ADC_SOC_Cnf.c, this function configures the ADC peripheral as specified in the arguments. For details on how and what arguments are passed, see the source file. This is a macro that is part of the TI PowerLib.

- **ADCDRV_1ch n**

  Defined in ADCDRV_1ch.asm, this Macro abstracts the usage of ADC module. The driver macro copies the result from the ADCRegisters into a NetBus array variable, which can be used to connect to various Library Macros. This is a macro that is part of the TI PowerLib.

The LED-ColorMix project has the following properties:

RAM Memory Usage of the LED-ColorMix Project		
Build Level	**Program Memory Usage 2802x**	**Data Memory Usage 2802x** (1)
Build 2 (FLASH)	594 words	949 words

(1) Excluding the stack size

CPU Utilization of Build 2 of the LED-ColorMix Project	
Name of Modules (1)	**Number of Cycles**
Context save, restore, and so forth	22
ADCDRV_1ch (1)4	27
Timeslicing overhead	11
LED control (four strings updated each ISR)	
ADCDRV_1ch (1) 4	27
CNTL_2P2Z (1) 4	192
PWMDRV_2ch_UpCnt (1) 2	31
Total Number of Cycles	310
CPU Utilization @ 60 Mhz	68.7% (2)

(1) The modules are defined in the header files as “macros”.
(2) At 133 kHz ISR frequency.

Table 4. System Features

System Features	
Development and Emulation	Code Composer Studio™ v4.1 (or above) with real-time debugging
Target Controller	TMS320F2802x
PWM Frequency	400 kHz PWM
PWM Mode	Up-count mode with independent output on A and B channels
Interrupts	EPWM1 on every third time base period event – implements 133 kHz ISR execution rate

5 Hardware Setup

In this guide, each component is named first with their macro number followed by the reference name. For example, [M2]-J1 would refer to the jumper J1 located in the macro M2 and [Main]-J1 would refer to the J1 located on the board outside of the other defined macro blocks. Figure 9 shows some of the major connectors and features of the Multi-DC/DC color LED board.

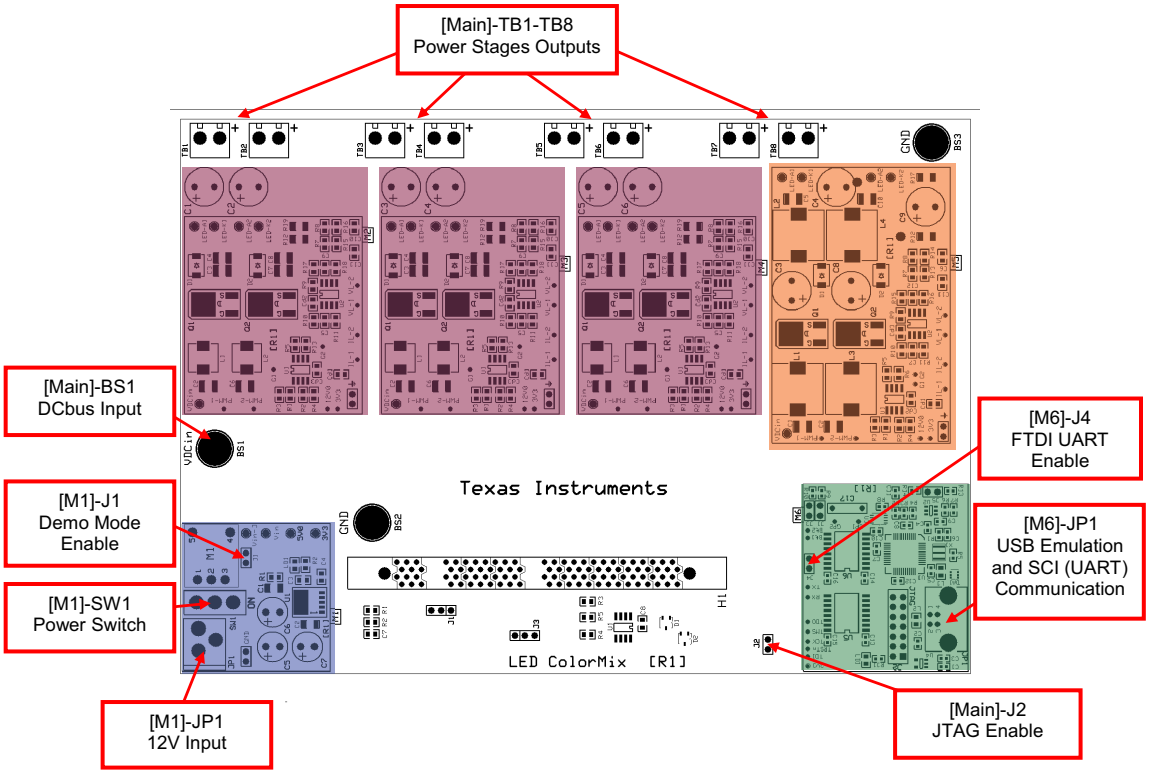


Figure 9. Hardware Features

[Main] – controlCARD socket, power connectors, and jumpers

[M1] – 12 V, 5 V, and 3.3 V DC power rails

[M6] – Isolated JTAG and serial communications interface (SCI) (universal asynchronous receiver/transmitter (UART)) communication via USB

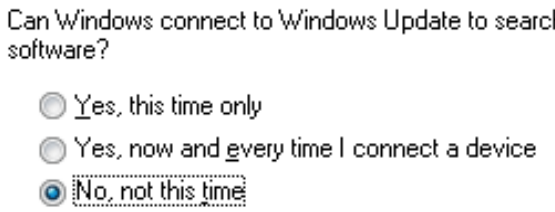
[M2],[M3],[M4] – Two independent boost stages, per macro instance

[M5] – Two independent sepic stages

1. Insert a F28027 control card into socket [Main]-H1.
2. Connect your computer to the board using a USB cable. [M6]-LD1 near the USB connector [M6]-JP1 should turn on.

NOTE: If Code Composer Studio has never been installed, it may be necessary to install drivers to make the board work correctly. If a popup comes up when the USB cable is connected from the board to the computer, have the install wizard install drivers from the XDS100v1 directory of the USB drive that are included with this kit.

3. When Windows asks to search Windows Update, select “No, not at this time” and click Next.



4. On the next screen select “Install from specific location” (see Figure 10) and click Next.

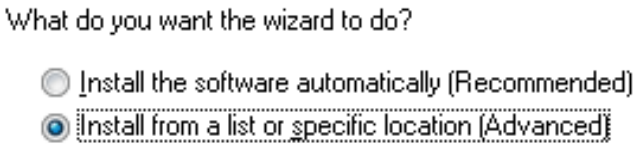


Figure 10. Install From Specific Location

5. Select “Search for Best Driver” (see Figure 11). Uncheck search removable media, and check include specific location and browse to [USB Drive]:\XDS100 Drivers.

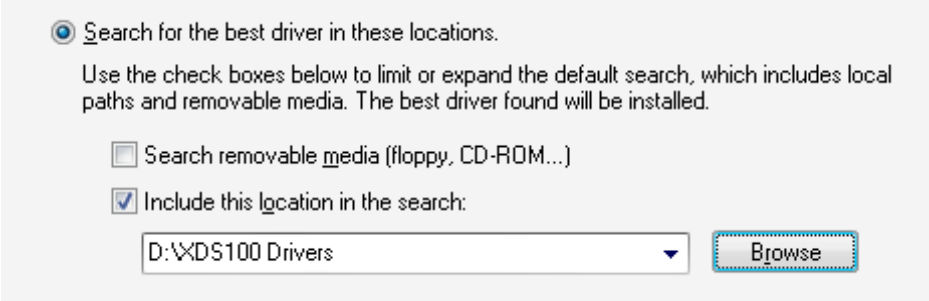


Figure 11. Search for Best Driver

6. Click next and the drivers will be installed. The driver install screen will appear three times, repeat this procedure each time.
7. Connect the LED panel to the board via [Main]-TB1-TB8.
8. Verify the following jumper settings:
   (a) No jumper is placed on [M1]-J1.
   (b) A jumper is placed on [M6]-J4.
   (c) A jumper is placed on [Main]-J2.
9. Connect the 12 V DC power supply to [M1]-JP1 to power the Auxiliary power rail.
10. Connect a 12 V – 18 V DC power supply’s output to [Main]-BS1 and Ground to [Main]-BS2/BS3 to power the Primary power rail.

6 Software Setup

6.1 Installing Code Composer Studio and controlSUITE

1. If not already installed, please install Code Composer Studio v4.x from the USB drive included in the kit.
2. Go to http://www.ti.com/controlsuite and run the controlSUITE installer. Select to install the “Multi-DC/DC Color LED Kit” software and allow the installer to also download all automatically checked software.

6.2 Setup Code Composer Studio to Work With the Multi-DC/DC Color LED Kit

1. Open “Code Composer Studio v4”.
2. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location. (Note that the workspace is a location on the hard drive where all the user settings for the IDE: which projects are open, what configuration is selected, where projects are saved, and so forth. This can be anywhere on the disk; the location mentioned below is just for reference. Also note that if this is not your first-time running Code Composer Studio, this dialog may not appear.)

   (a) Click the “Browse…” button.

   (b) Create the path below by making new folders as necessary.

   (c) “C:\Documents and Settings\<username>\My Documents\CCSv4_workspaces\LED-ColorMix”.

   (d) Uncheck the box that says “Use this as the default and do not ask again”.

   (e) Click “OK”.

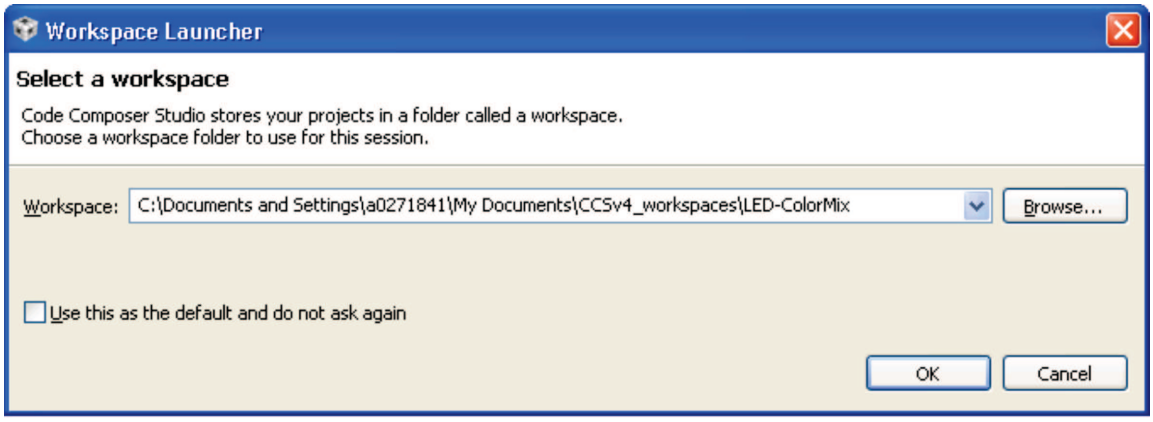


Figure 12. Workspace Launcher

3. Configure Code Composer Studio to know which MCU it will be connecting to. Click “Target → New Target Configuration…”. Name the new configuration “XDS100 F28027.ccxml”. Make sure that the “Use shared location” checkbox is checked and then click “Finish”.

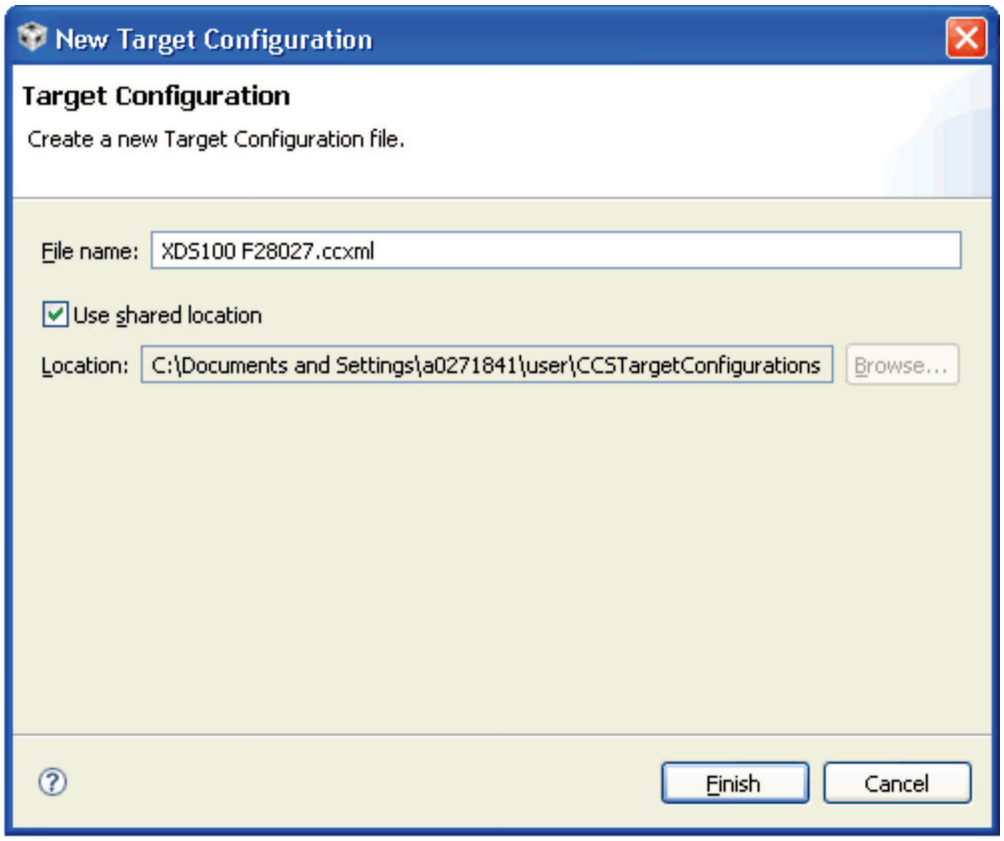


Figure 13. Creating a Target Configuration

4. This should open up a new tab as seen in Figure 14. Select and enter the options as shown:
   (a) Connection – Texas Instruments XDS100v1 USB Emulator.
   (b) Device – TMS320F28027.
   (c) Click Save.
   (d) Close the "XDS100 F28027.ccxml" tab.

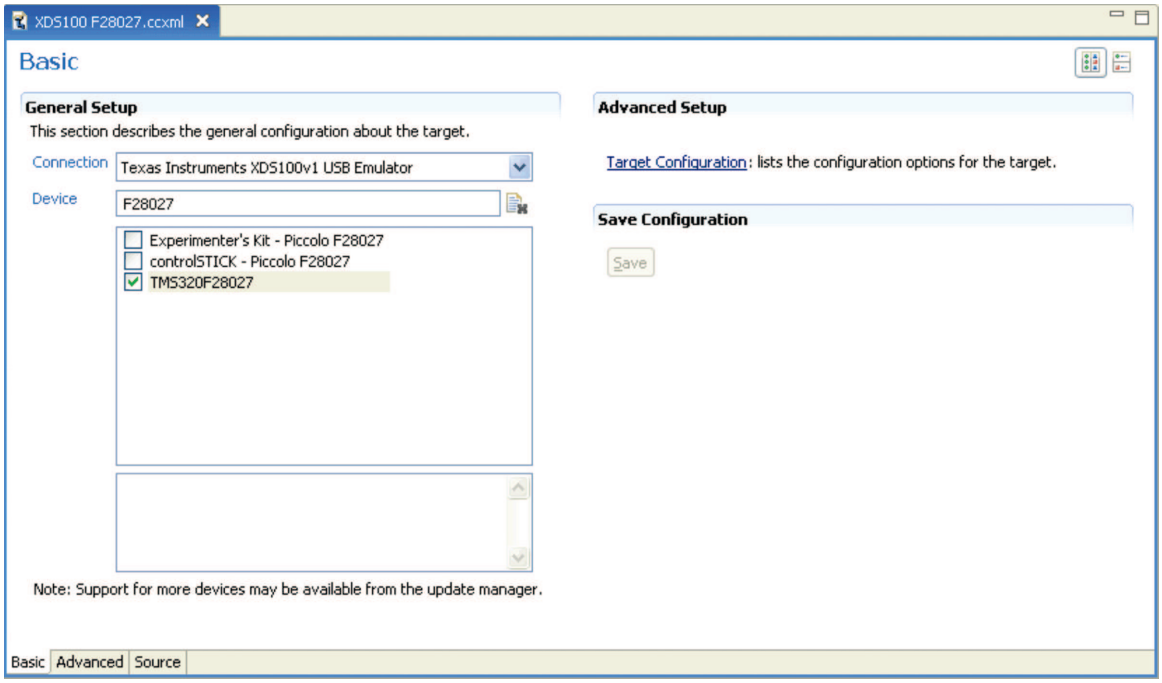


Figure 14. Configuring a New Target

5. Assuming this is your first time using Code Composer, the “XDS100 F28027” configuration is now set as the default target configuration for Code Composer Studio. Check this by going to “View → Target Configurations”. In the “User Defined” section, right-click on the “XDS00 F28027.ccxml” file and select “Set as Default”. This tab also allows you to reuse existing target configurations and link them to specific projects.

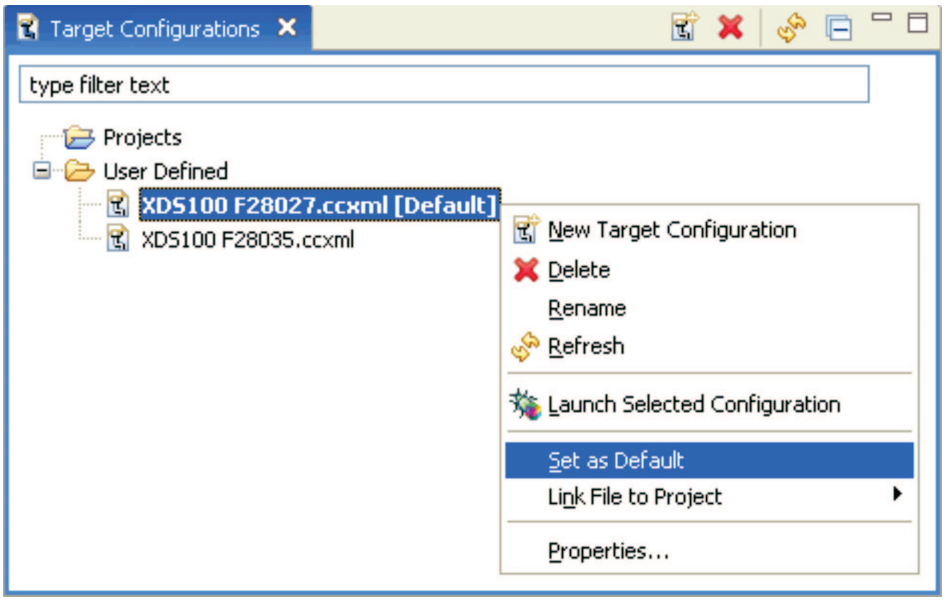


Figure 15. Setting a Default Target Configuration

6. Add the LED-ColorMix project into your current workspace by clicking “Project → Import Existing CCS/CCE Eclipse Project”.
   (a) Browse to the LED-ColorMix project directory within the Multi-DC/DC Color LED Kit folder. This will be: “C:\TI\controlSUITE\development_kits\Multi-DCDC-Color-LED-Kit_v1.0\”.
   (b) Click “Finish” to load the LED-ColorMix project into the workspace.

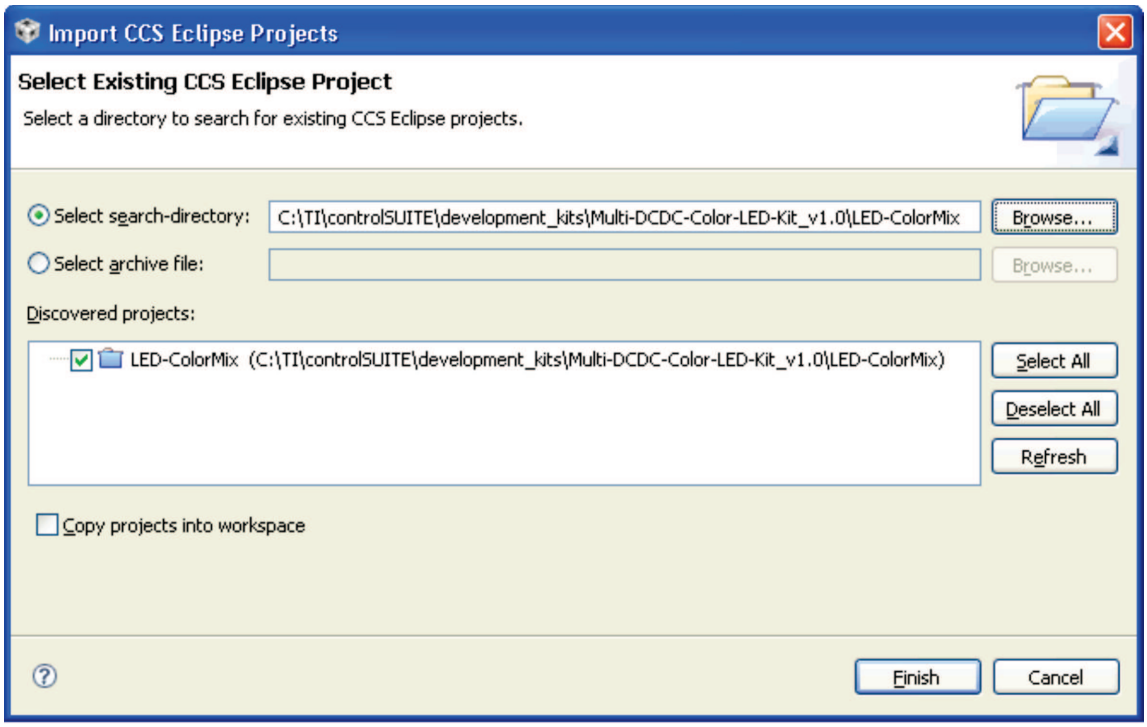


Figure 16. Importing the Multi-DC/DC Color LED Kit Project to Your Workspace

7. The LED-ColorMix project should be set as the active project. Right-click on the project name and click "Set as Active Project". Expand the file structure of the project.

6.3 *Incremental System Build for LED-ColorMix Project*

The lighting system is gradually built up in order for the final system to be operated confidently. Two phases (system builds) are designed to verify the major software modules used in the system:

- Build 1: Open Loop Test, check basic operation of the DC/DC stages and LED strings
- Build 2: Closed Loop operation of the DC/DC stages and LED strings

Table 5. Testing Modules in Each Incremental System Build [(1)]

Software Module	Phase 1	Phase 2
PWMDRV_2ch_UpCnt	√√	√
ADCDRV_1ch	√√	√
CNTL_2P2Z		√√

(1) The √ symbol means this module is used in this phase and the √√ symbol means that this module is tested in this phase.

7 Build 1: Open Loop Operation of the DC/DC Stages and LED Strings

NOTE: This section assumes that Section 5 and Section 6 have been completed. If not, go through these sections before continuing.

The objective of this build is as follows:

- Verify that the power stages on the board are working correctly.
- Control the duty cycles of the various power stages on the board and evaluate their output.

The components of the system as used in the software are described in Figure 17.

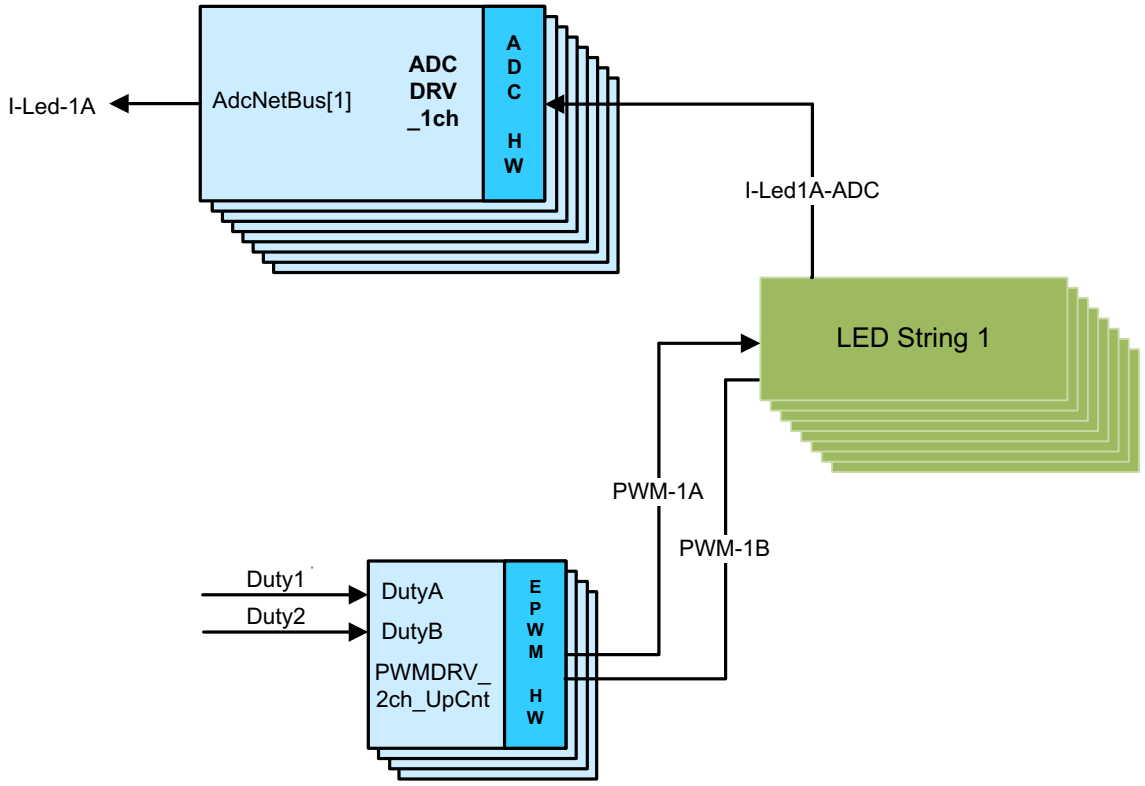


Figure 17. Build Level 1 Block Diagram

7.1 Inspect the Project

After the initial boot processes are complete, the software begins from the main function.

1. Open up LED-ColorMix-Main.c and find the *main()* function (line 372). The first thing that the software does in the *main()* function is call a function called *DeviceInit()* found in LED-ColorMix-DevInit_F2802x.c.
2. Open and inspect LED-ColorMix-DevInit_F2802x.c by double clicking on the filename in the project window. In this file, the various peripheral clocks are enabled or disabled and the functional pinout, that configures which peripherals come out of which pins, is defined.
   (a) Confirm that the ADC and PWM1-4 peripheral clocks are enabled (lines 99-117). Also confirm that GPIO00-GPIO07 are configured to be PWM outputs (lines 136-182).
3. The LED-ColorMix project is provided with incremental builds where different components and macro blocks of the system are pieced together one by one to form the entire system. This helps in step-by-step debug and understanding of the system.
   (a) From the C/C++ Project tab, open the file LED-ColorMix-Settings.h and make sure that INCR_BUILD is set to 1 and save this file. After you test build 1, this variable needs to be redefined to move on to build 2, and so on until all builds are complete.
4. Go back to the LED-ColorMix-Main.c file and notice the various incremental build configuration code. The Build LEVEL1 configuration code is found on lines 595-630. In it the ADC, PWM and macro block connections are configured. Note that if INCR_BUILD is set to 1 in LED-ColorMix-Settings.h, lines 634-710 will be ignored by the compiler because they do not belong in the current build. A similar method of enabling or disabling code based on the value of INCR_BUILD is done in LED-ColorMix-ISR.asm as well.

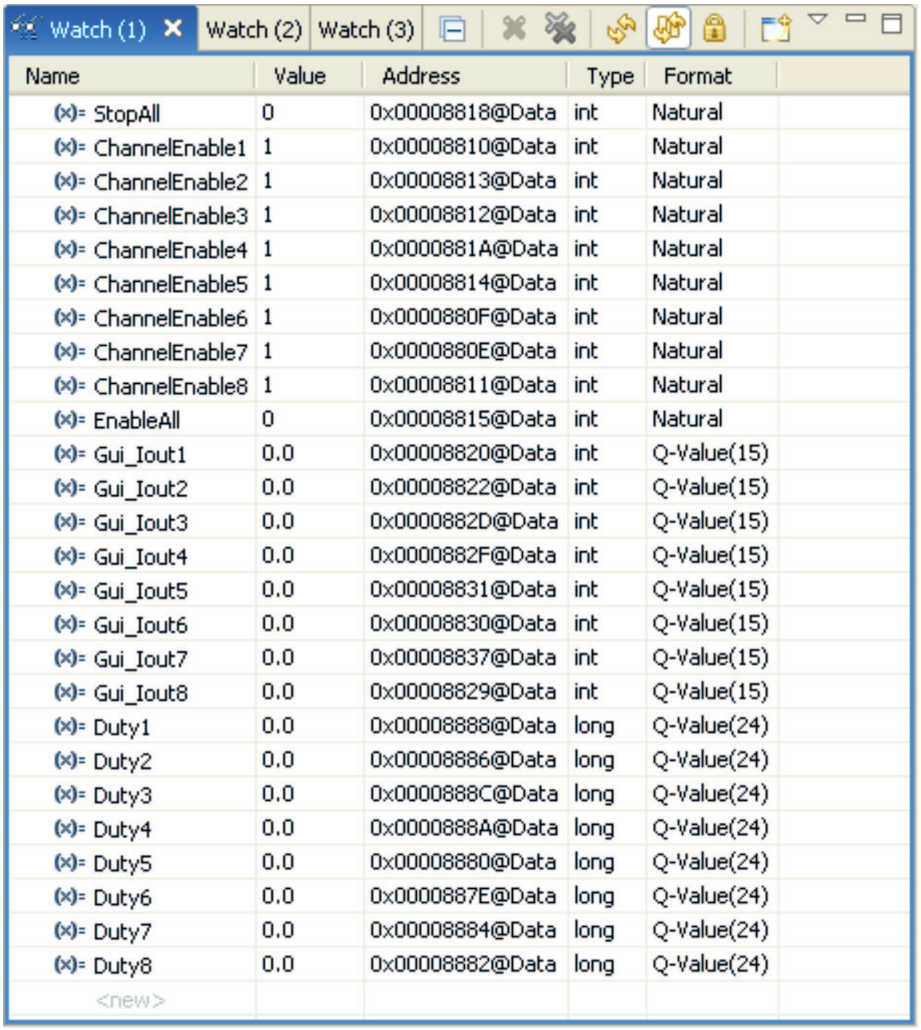
7.2 Build and Load the Project

1. Right Click on the Project Name and click on “Rebuild Project”; watch the Console window. Any errors in the project are displayed in the Console window.
2. On successful completion of the build click the “Debug” button  , located in the top-left side of the screen. The IDE will now automatically connect to the target, load the output file into the device and change to the Debug perspective.
3. Now click the real-time mode button  that says “Enable silicon real-time mode”. This will allow you to edit and view variables in real-time without halting the program.
4. A message box may appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.

7.3 Setup Watch Window and Graphs

1. Click: View → Watch on the menu bar to open a *watch window* to view the variables being used in the project. Add the variables found in Figure 18 to the watch window. The format of a variable can be changed by right-clicking on a particular variable then selecting a Q-Value. Change the format of each variable to match the figure. The variables in this watch window are used to control and monitor the status of the board while in Build 1.

   Hint: Shift-Click can be used select multiple variables or elements and then right-clicking and changing the format will affect all variables selected.



Name	Value	Address	Type	Format
StopAll	0	0x00008818@Data	int	Natural
ChannelEnable1	1	0x00008810@Data	int	Natural
ChannelEnable2	1	0x00008813@Data	int	Natural
ChannelEnable3	1	0x00008812@Data	int	Natural
ChannelEnable4	1	0x0000881A@Data	int	Natural
ChannelEnable5	1	0x00008814@Data	int	Natural
ChannelEnable6	1	0x0000880F@Data	int	Natural
ChannelEnable7	1	0x0000880E@Data	int	Natural
ChannelEnable8	1	0x00008811@Data	int	Natural
EnableAll	0	0x00008815@Data	int	Natural
Gui_Iout1	0.0	0x00008820@Data	int	Q-Value(15)
Gui_Iout2	0.0	0x00008822@Data	int	Q-Value(15)
Gui_Iout3	0.0	0x0000882D@Data	int	Q-Value(15)
Gui_Iout4	0.0	0x0000882F@Data	int	Q-Value(15)
Gui_Iout5	0.0	0x00008831@Data	int	Q-Value(15)
Gui_Iout6	0.0	0x00008830@Data	int	Q-Value(15)
Gui_Iout7	0.0	0x00008837@Data	int	Q-Value(15)
Gui_Iout8	0.0	0x00008829@Data	int	Q-Value(15)
Duty1	0.0	0x00008888@Data	long	Q-Value(24)
Duty2	0.0	0x00008886@Data	long	Q-Value(24)
Duty3	0.0	0x0000888C@Data	long	Q-Value(24)
Duty4	0.0	0x0000888A@Data	long	Q-Value(24)
Duty5	0.0	0x00008880@Data	long	Q-Value(24)
Duty6	0.0	0x0000887E@Data	long	Q-Value(24)
Duty7	0.0	0x00008884@Data	long	Q-Value(24)
Duty8	0.0	0x00008882@Data	long	Q-Value(24)
<new>				

Figure 18. Configuring the Watch Window for Build 1

2. Click on the Continuous Refresh button  in the watch window. This enables the window to run with real-time mode. By clicking the down arrow in any watch window, you may select “Customize Continuous Refresh Interval” and edit the refresh rate for the watch windows. Note that choosing too fast an interval may affect performance.

7.4 Run the Code

1. Run the code by pressing Run Button  in the Debug Tab.
2. The project should now run, and the values in the watch window should keep on updating. You may want to resize or reorganize the windows according to your preference. This can be done easily by dragging and docking the various windows.
3. With the primary power supply set to 0 V or turned off, verify the PWM and ADC circuitry:
   (a) Set Duty[1-8] to 0.50 which corresponds to 50% duty cycle. Verify by scoping the PWM test points located on the board.
   (b) Verify that all Voltage and Current feedback variables are close to 0 (less than 0.005). They may not be exactly 0.0 as it is normal for some low level noise to be present on the ADCs.
4. Set Duty[1-8] to 0.0 once complete.

5. Set Duty1 to 0.57, with the primary power supply set to 12 V. This sets the duty cycle of the MOSFET controlling LED string 1 to be 57%. Verify that LED string 1 turns on and that Gui_Iout1 changes to about 0.010. Carefully adjust the Duty1 and verify that Gui_Iout1 increases or decreases when you increase or decrease Duty1. Be careful not to exceed the board's current ratings.
6. LED strings 2-8 can be checked by adjusting Duty[2-8] and monitoring Gui_Iout[2-8], respectively. Use the following Duty{X} values and verify the corresponding Gui_Iout{X}:
   (a) Duty2: 0.59, Gui_Iout2: 0.010
   (b) Duty3: 0.40, Gui_Iout3: 0.010
   (c) Duty4: 0.57, Gui_Iout4: 0.010
   (d) Duty5: 0.59, Gui_Iout5: 0.010
   (e) Duty6: 0.40, Gui_Iout6: 0.010
   (f) Duty7: 0.33, Gui_Iout7: 0.010
   (g) Duty8: 0.33, Gui_Iout8: 0.010
7. Once complete, set Duty[1-8] to 0.0.
8. Disable real-time mode  and Reset the processor  (Target → Reset → Reset CPU), and then terminate the debug session by clicking  (Target → Terminate All). This halts the program and disconnects Code Composer Studio from the MCU.

8 Build 2: Closed Loop Operation of the DC/DC Stages and LED Strings

The objective of this build is as follows:

Regulate each LED string's current draw using average current mode control with closed-loop feedback.

The components of the system as used in the software are described in Figure 19.

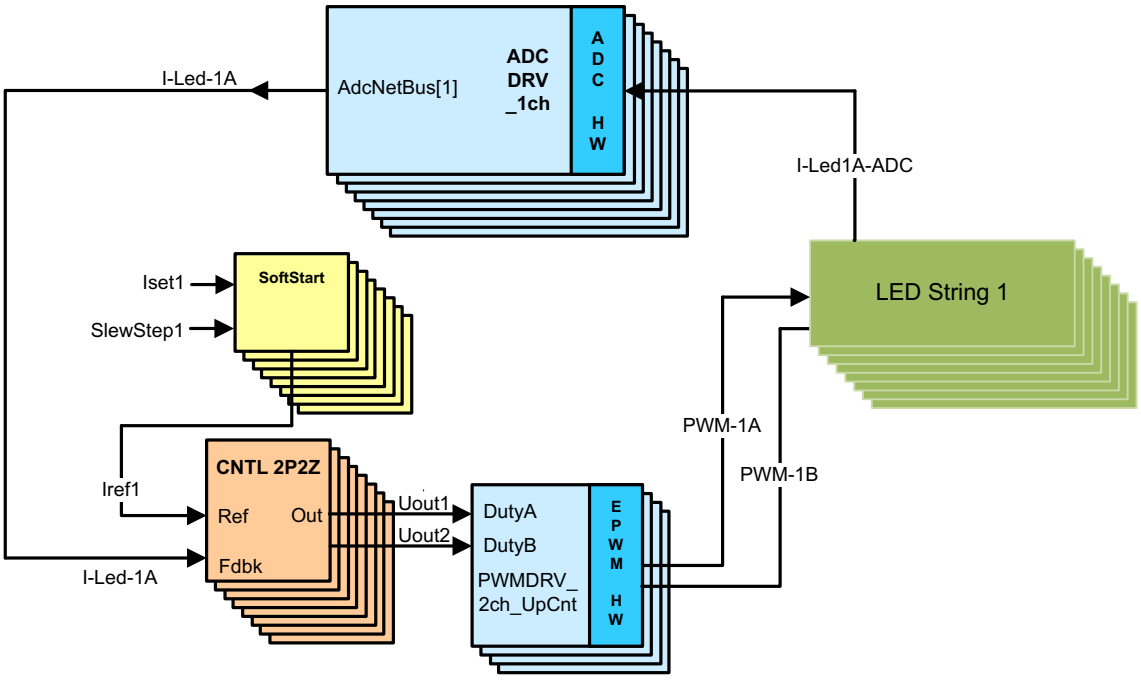


Figure 19. Build Level 2 Block Diagram

The instructions for this build are more succinct than in the first build. For further guidance, see the instructions in Section 7. Run Build 1 prior to running Build 2.

8.1 Build and Load the Project

1. Open LED-ColorMix-Settings.h and change the incremental build level to 2 (#define INCR_BUILD 2). Save the file.
2. Right-click on the project name and select “Rebuild Project”.
3. On successful completion of the build click the “Debug” button  , located in the top-left side of the screen. The IDE will now automatically connect to the target, load the output file into the device and change to the Debug perspective.
4. Click the real-time mode button  that says “Enable silicon real-time mode”. This allow you to edit and view variables in real-time without halting the program.

8.2 Setup Watch Window and Graphs

1. Add a new watch window to the workspace and add the variables, and set it to use the correct format as shown in Figure 20. The variables in this watch window are used to control and monitor the status of the board while in Build 2.

Watch (1) | Watch (2) | Watch (3)

Name	Value	Address	Type	Format
StopAll	0	0x00008818@Data	int	Natural
ChannelEnable1	1	0x00008810@Data	int	Natural
ChannelEnable2	1	0x00008813@Data	int	Natural
ChannelEnable3	1	0x00008812@Data	int	Natural
ChannelEnable4	1	0x0000881A@Data	int	Natural
ChannelEnable5	1	0x00008814@Data	int	Natural
ChannelEnable6	1	0x0000880F@Data	int	Natural
ChannelEnable7	1	0x0000880E@Data	int	Natural
ChannelEnable8	1	0x00008811@Data	int	Natural
EnableAll	0	0x00008815@Data	int	Natural
Gui_Iout1	0.0	0x00008820@Data	int	Q-Value(15)
Gui_Iout2	0.0	0x00008822@Data	int	Q-Value(15)
Gui_Iout3	0.0	0x0000882D@Data	int	Q-Value(15)
Gui_Iout4	0.0	0x0000882F@Data	int	Q-Value(15)
Gui_Iout5	0.0	0x00008831@Data	int	Q-Value(15)
Gui_Iout6	0.0	0x00008830@Data	int	Q-Value(15)
Gui_Iout7	0.0	0x00008837@Data	int	Q-Value(15)
Gui_Iout8	0.0	0x00008829@Data	int	Q-Value(15)
Gui_Iset1	0.0	0x00008824@Data	int	Q-Value(15)
Gui_Iset2	0.0	0x00008827@Data	int	Q-Value(15)
Gui_Iset3	0.0	0x00008826@Data	int	Q-Value(15)
Gui_Iset4	0.0	0x0000882E@Data	int	Q-Value(15)
Gui_Iset5	0.0	0x0000881E@Data	int	Q-Value(15)
Gui_Iset6	0.0	0x0000881D@Data	int	Q-Value(15)
Gui_Iset7	0.0	0x0000881F@Data	int	Q-Value(15)
Gui_Iset8	0.0	0x00008821@Data	int	Q-Value(15)
<new>				

Figure 20. Configuring the Watch Window for Build 2

2. Click on the Continuous Refresh button  in the watch window.

8.3 Run the Code

1. Run the code by pressing Run Button  in the Debug Tab.
2. The project should run and the values in the watch window should keep on updating. You may want to resize or reorganize the windows according to your preference. This can be done easily by dragging and docking the various windows.
3. In the watch window for Build 2, set the variable Gui_Iset1 to 0.01, which corresponds to 0.010A. When the current reference is changed, the value of Gui_Iout1 should change to match the reference value of 0.01.
4. Change the value of Gui_Iset1 to 0.03. Notice that the LED string brightness has increased with the increased current.
5. Continue editing Gui_Iset1 and also Gui_Iset[2-8] with various values from 0.0 to 0.100.

NOTE: Do not increase the output current by each string above 0.100A. This is because the Boost stage inductors are only designed to handle 0.350A of current. If a larger amount of output current is required, replace the Boost inductors with those of a higher current rating.

NOTE: Different colors can be produced by changing the current ratio between the Red, Green, and Blue LED strings.

6. Once complete, set Gui_Iset[1-8] to 0.0 or set StopAll to 1.
7. Disable real-time mode  and Reset the processor  (Target → Reset → Reset CPU), and then terminate the debug session by clicking  (Target → Terminate All). This halts the program and disconnects Code Composer Studio from the MCU.

9 Ideas on Further Exploration

- I2C Temperature Sensing:

  A temperature sensor could be placed on the LED panel in order to give the system information on the temperature of the LEDs on the panel. This temperature information could then be sent to an external host, be used to provide thermal protection for the LEDs, or (along with a lookup table) be used to provide thermal compensation.
- LED current offset compensation:

  Resistors, op-amps and other discrete components have a specified error inherent in their design. These errors often manifest themselves as ADC voltage offset errors in a system. To reduce this variablility and error, the ADC voltage could be measure a voltage at a known state and derive the error. The digital controller could then subtract this error within the interrupt.
- Communication Protocol:

  Many lighting systems also take advantage of communication protocols to increase features and improver performance and efficiency. Lighting systems use communication such as DMX512, DALI, and even Power Line Communication (PLC). An experimental DMX512 circuit is included on the board as an example.
- Use the Multi-DC/DC Color LED Kit to Jump-Start Your Prototype:

  All software and hardware that is bundled with this kit is free to use with no licenses. Feel free to use any of the hardware and software as resources for your own design.

10 References

For more information, see the following documents that are located at: www.ti.com/controlsuite:

- LED-ColorMix-AppNote – provides detailed information on the CCS4 project.
  \TMDSRGBLEDKIT_v1.0\~Docs\LED-ColorMix-AppNote[R1].pdf

- LED-ColorMix-HWGuide – provides detailed information on the hardware on the board.
  \TMDSRGBLEDKIT_v1.0\~Docs\LED-ColorMix-HWGuide[R1].pdf
- LED-ColorMix-HWdevPkg – a folder containing files related to the hardware on the board (schematics, bill of materials, Gerber files, PCB layout, and so forth).
  \TMDSRGBLEDKIT_v1.0\~LED-ColorMix-HWdevPkg[R1]\

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

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		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265