

# TI Designs MSP430™ Software RGB LED Control Design Guide



## TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help you accelerate your time to market.

## Design Resources

- [TIDM-G2XXSWRGBLED](#) Tool Folder Containing Design Files
- [MSP430G2553](#) Product Folder
- [MSP-EXP430G2](#) Product Folder



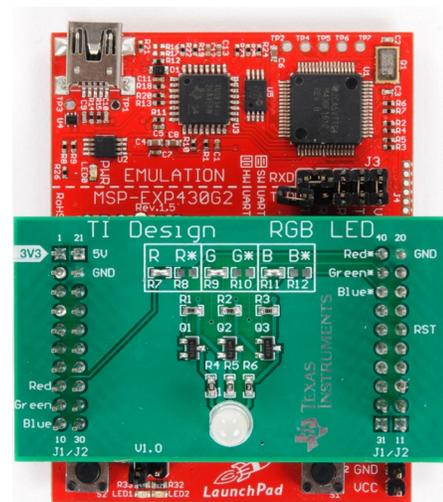
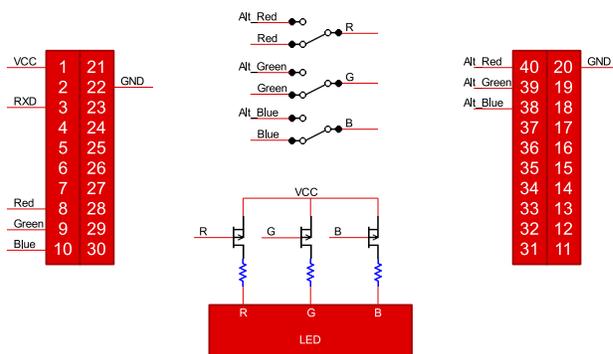
[ASK Our E2E Experts](#)  
[WEBENCH® Calculator Tools](#)

## Design Features

- Red, Green, and Blue (RGB) LED Control With P-FETs to Source LED Current Directly From VCC
- Independent Control of Color and Brightness
- 512 Unique Colors and 50 Brightness Levels
- Software PWM Generation Uses a Single Compare Capture Register (CCR) on One Timer Module
- Serial Interface for Externally Controlling LED

## Featured Applications

- Multi-Color LED Indication



An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

All trademarks are the property of their respective owners.

## 1 System Description

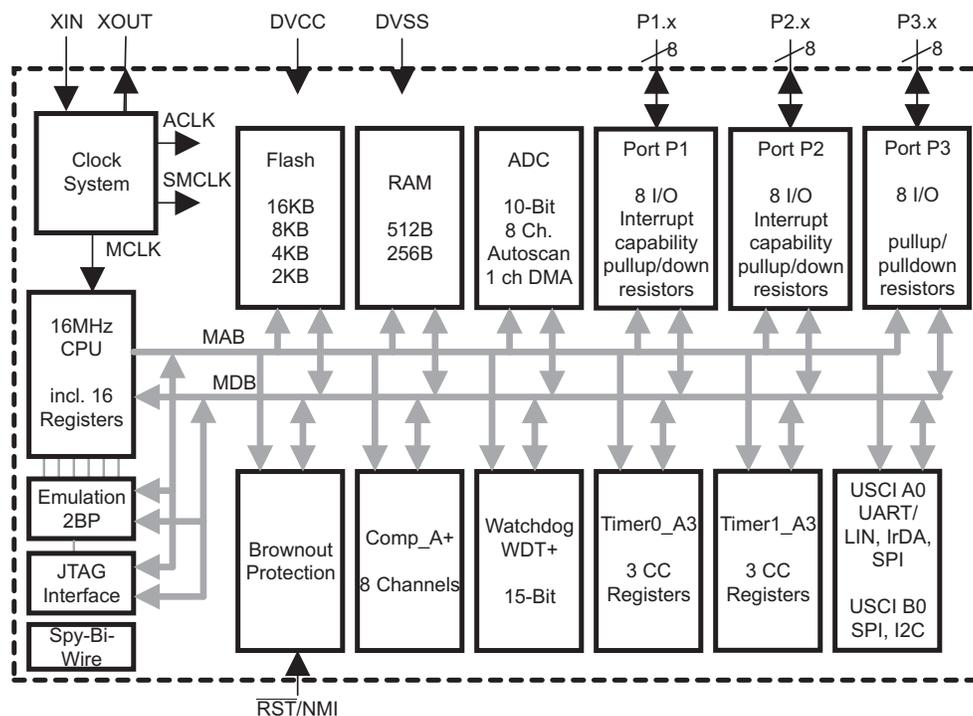
This reference design describes the implementation of an RGB tri-color LED with the following characteristics:

- RGB LED control with P-FETs to source LED current directly from VCC
- Independent control of color and brightness
- A single timer and CCR module to work on devices without sufficient peripherals to create three pulse-width modulation (PWM) signals in hardware
- Serial interface to externally control the RGB LED
- Works on any MSP430 operating at MCLK = 8 MHz or higher (a software example for MSP430G2553 is included)

### 1.1 MSP430G2553

The Texas Instruments™ MSP430G2553 is an ultra-low-power microcontroller from the MSP430 family of devices. Combined with extensive low-power modes, the architecture is optimized to achieve extended battery life in portable applications. The device features a powerful 16-bit RISC CPU, 16-bit registers, and mixed signal integration.

Devices in the MSP430G2xx3 family have two 16-bit timers, an optional 10-bit ADC with eight input channels (G2x53 devices), one universal serial communications interface (USCI), a comparator, and up to 24 I/O pins. This reference design uses the MSP430G2553 in the MSP-EXP430G2 LaunchPad™, which is the superset of the G2xx3 family of devices, with 16KB flash, 512 bytes RAM, and 16 I/Os in a DIP package.



**Figure 1. MSP430G2553 Functional Block Diagram**

## 2 Block Diagram

Figure 2 shows the block diagram for the BoosterPack™ version of this design. The three-color PWM pins have alternate pin locations on the 40-pin BoosterPack connector. These locations are denoted with an \* on the BoosterPack and "Alt" in the following block diagram.

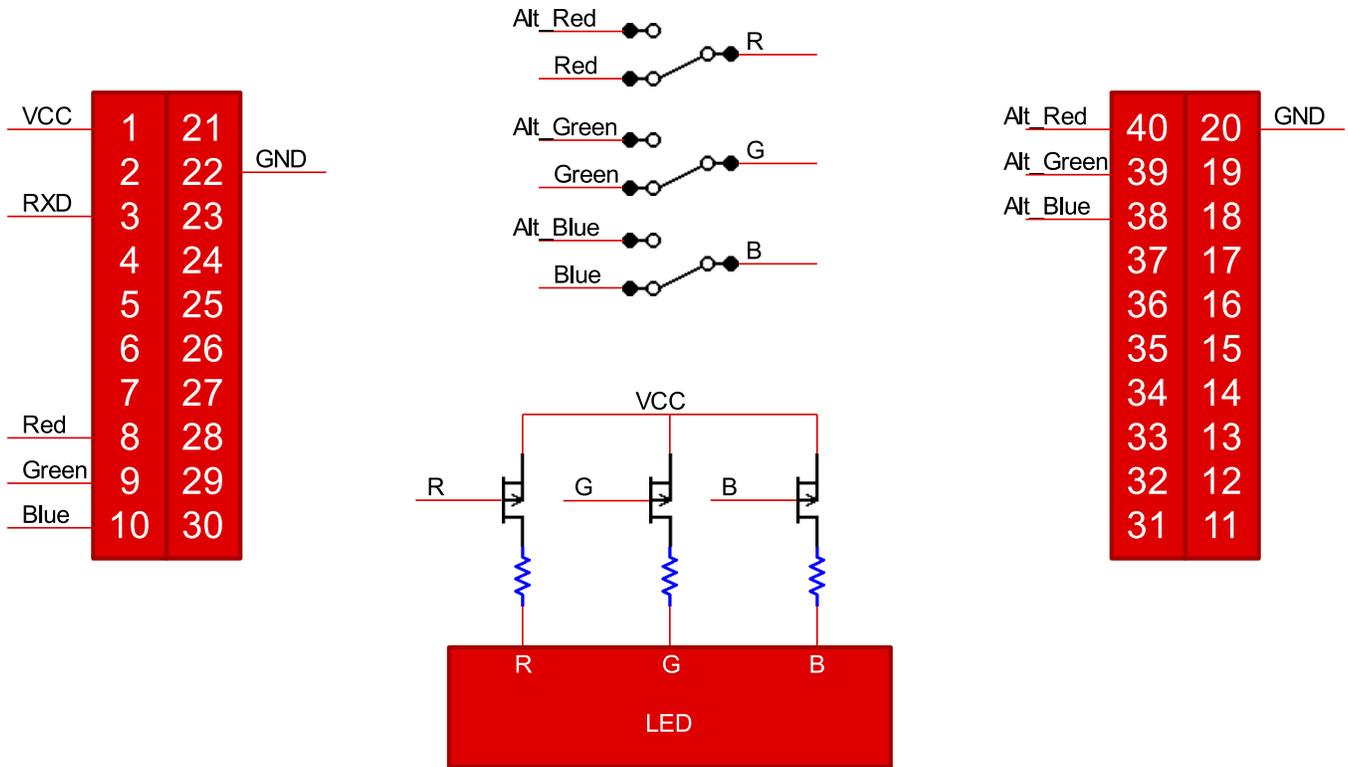


Figure 2. System Block Diagram

### 3 System Design Theory

#### 3.1 RGB Color Mixing

The output color of the RGB LED is controlled by varying the level of each of the three-color LEDs: red, green, and blue. The three colors are combined to create an additive color result. The level of each color is set by the duty cycle of the PWM signal driving the P-FET controlling the LED. As long as the frequency of the PWM is higher than 50 Hz, no flickering is observable to human eyes. Higher duty cycles yield higher levels of each color.

Figure 3 shows three PWM signals (for red, green, and blue) with different duty cycles, resulting in different levels for each color in the RGB LED. The red PWM is 50% duty cycle, while the green is 100%, and the blue is 33%. The result is an additive color that is light green.

---

**NOTE:** In the waveforms shown in this section, a high value corresponds to LED on and a low value to LED off. The actual PWMs to control the P-FET in the reference circuit provided will be the inverse: output low corresponds to LED on and output high to LED off.

---

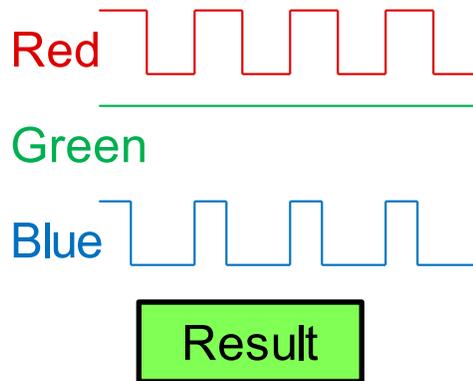
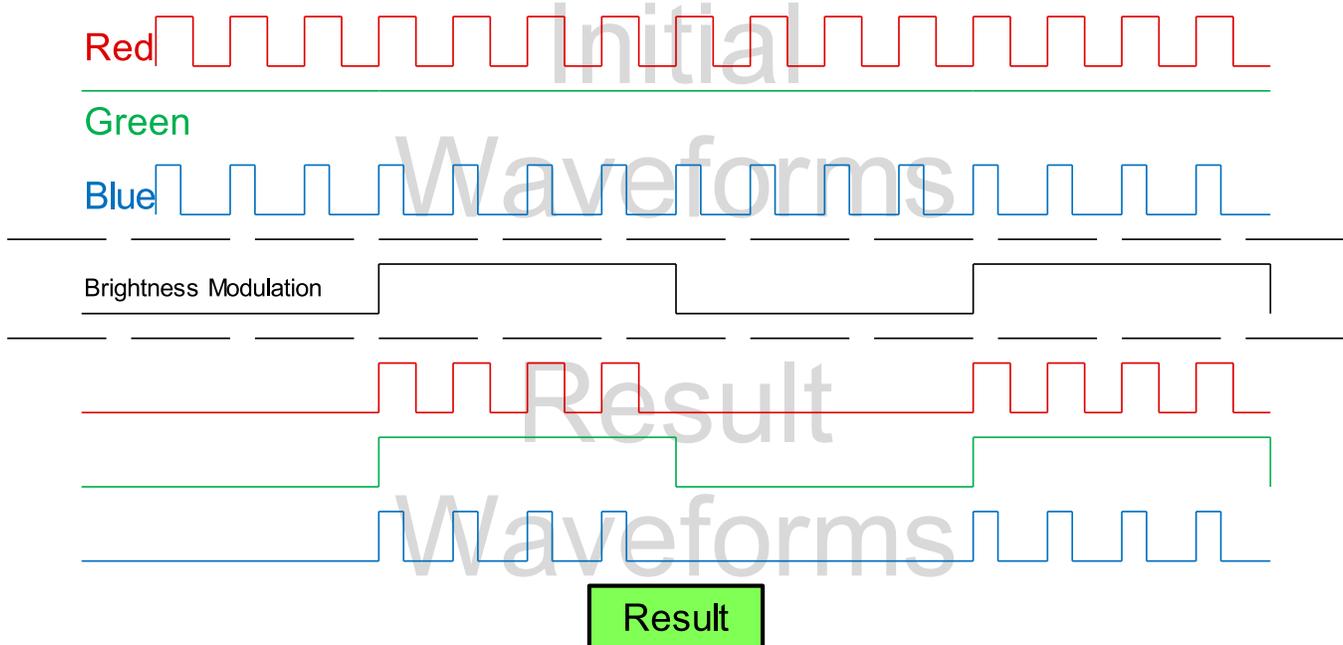


Figure 3. PWM Color Mixing

The overall brightness of the RGB LED is adjusted by pulse-width modulating the PWM signals for each color, where the overall modulation is the same for all three PWM signals. This allows the color levels to remain the same ratio between red, green and blue, while dimming or increasing the overall brightness. To prevent flickering of the LED, the brightness modulation should also be faster than 50 Hz. Figure 4 shows the same three PWM signals as above for initial wave forms with a brightness modulation added. The resulting waveforms create a color that is the same additive color as Figure 3, but the overall brightness of the RGB LED will be significantly reduced.



**Figure 4. Brightness Adjusted PWM Color Mixing**

Note that for typical LEDs, the duty cycle of the PWM does not perfectly match to the output level of the LED. However, for simple applications such as this design, the relationship is assumed to be linear.

### 3.2 LED Control

A logic-level P-channel MOSFET is used as a switch for each color channel to turn the individual LEDs on and off. Using a P-FET allows the LEDs to source current directly from the VCC power rail instead of the MSP430, protecting the MSP430 from the high current some LEDs can draw.

The source of the P-FET is connected to VCC and the drain to the LED current limiting resistor, then to the anode of the LED itself. The PWM output of the MSP430 is connected to the gate of the P-FET to allow current to flow through the FET, enabling and disabling the LED. In this configuration, a high voltage on the gate disables the LED and a low voltage enables the LED. A 1-M $\Omega$  resistor connects the gate to the source to provide a very weak pull-up on the control logic, disabling the LED when the MSP430 is not actively driving a signal. This circuit is repeated for each of the three colors in the RGB LED.

In most RGB LEDs, the different colors have different diode voltages and current to brightness relationships. Use the RGB LED datasheet and take care to select current limiting resistors for each color to match the brightness level of the three LEDs. Otherwise, the resulting additive color will not be correct. Section 4.1 provides more information on selecting an RGB LED and the appropriate current limiting resistors.

### 3.3 Creating PWMs in Software

Creating three PWMs on a device without sufficient timer CCR registers to create hardware PWMs (such as many smaller MSP430s) can be done using a software PWM algorithm. Traditionally, software PWMs are created using a software counter that increments at a constant frequency and toggling a GPIO pin at two-count values, resetting the counter at the higher value. Changing the smaller value (relative to the larger) changes the duty cycle of the PWM and changing the larger value changes the frequency of the PWM. Additional PWMs can be created of the same frequency by comparing the count to additional smaller values to toggle different GPIO pins. When the counter is reset, all the PWM GPIO pins used are toggled.

To create the brightness modulated PWMs described in this design, two software counters are used. The first counter, called a level counter in this design, is used in the traditional software PWM algorithm (as described above) and increments the second counter, called the brightness counter in this design, every time the level count resets. The brightness counter is also compared to two-count values. When the smaller value of the brightness counter is reached, all the PWM GPIO pins are forced to the LED OFF state, regardless of the expected output of the level counter, and kept there until the larger brightness count value is reached and the brightness count is reset. Once the brightness count is reset, the PWM GPIO pins are again controlled by the level counter.

Figure 5 shows the process for creating the waveform shown above in Figure 4. Both counters, Level and Brightness (L and B, respectively), start at 0. The level counter counts to 5, resetting on the next count, giving six steps. The brightness counter counts to 7, resetting on the next count giving eight steps. The red level is set to three out of six, the green to six out of six, and blue to two out of six. The brightness modulation is set to four out of eight. During the level counter interval, the red LED is enabled until the level count reaches 3, the green LED is always enabled, and the blue LED is enabled until the level count reaches 2. When the level counter resets, the brightness counter increments and the LEDs are turned back on again. When the brightness count reaches 4 (after four complete level count periods), all three LEDs are disabled, regardless of level count value, until the brightness count resets back to 0. The resulting additive color is the same green with reduced brightness.

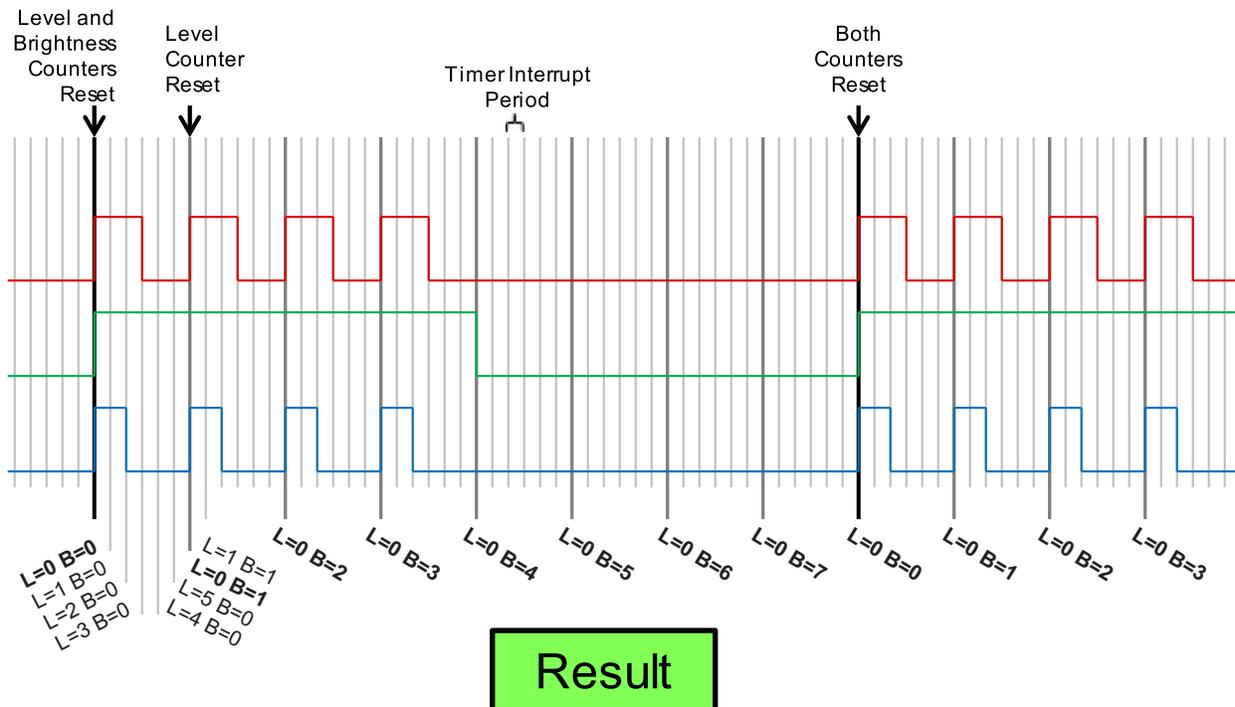


Figure 5. Software Generated PWM Signals

### 3.4 RGB LED Effects

RGB LED lighting effects can be implemented in software to enable shorter and fewer commands over the serial interface. This design contains the following implemented effects:

- Fade the RGB LED from one brightness level to another at a specified rate while maintaining a specified color
- Continuously fade the RGB LED from one brightness level to another then back to the first level at a specified rate while maintaining a specified color

Implementing the effects requires controlling the rate at which the RGB LED brightness level is changed. The software PWM algorithm includes a wake-up counter that can be used to wake up the MSP430 from low power mode after a specified number of timer interrupts. This counter allows the effects to be implemented at the required timings.

### 3.5 Serial Interface

A serial interface is included in the design to allow an external controller to send commands to the MSP430. The command packet structure is set up as follows:

- One byte: Length — Total length of the packet, including the length byte
- One byte: Command — Specifies which command is being sent
- n bytes: Command Data — Any additional data required by the specific command

This design has the following commands implemented:

- 0x00 — Stop an active command and turn off the RGB LED
  - Length — 0x02
  - Command — 0x00
  - Command data:
    - (none)
- 0x01 — Set the RGB LED to a specific color and brightness
  - Length — 0x06
  - Command — 0x01
  - Command data:
    - Red level — (one byte)
    - Green level — (one byte)
    - Blue level — (one byte)
    - Brightness level — (one byte)
- 0x02 — Fade the RGB LED
  - Length — 0x09
  - Command — 0x02
  - Command data:
    - Red level — (one byte)
    - Green level — (one byte)
    - Blue level — (one byte)
    - Initial brightness level — (one byte)
    - Final brightness level — (one byte)
    - Total time of effect (in ms) — (two bytes, MSB first)

- 0x03 — Continuously pulse the RGB LED
  - Length — 0x09
  - Command — 0x03
  - Command data:
    - Red level — (one byte)
    - Green level — (one byte)
    - Blue level — (one byte)
    - Low brightness level — (one byte)
    - High brightness level — (one byte)
    - Total time of single pulse, low-to-high-to-low, (in ms) — (two bytes, MSB first)

For example, the following commands are decoded:

- 0x06, 0x01, 0x08, 0x08, 0x00, 0x32
  - 0x06 — The command packet is six bytes long.
  - 0x01 — Command: Set the RGB LED to a specific color and brightness.
  - 0x08 — Set the color level of Red to 8, which is maximum in this design.
  - 0x08 — Set the color level of Green to 8, which is maximum in this design.
  - 0x00 — Set the color level of Blue to 0, or off.
  - 0x32 — Set the brightness to 50 (0x32), which is maximum in this design.
  - The command sets the Red to full on, Green to full on, and Blue to off, creating a yellow additive color. The brightness is set to full.
- 0x09, 0x03, 0x08, 0x04, 0x00, 0x05, 0x2D, 0x07, 0xD0
  - 0x09 — The command packet is nine bytes long.
  - 0x03 — Command: Continuously pulse the RGB LED.
  - 0x08 — Set the color level of Red to 8, which is maximum in this design.
  - 0x04 — Set the color level of Green to 4, which is 50% in this design.
  - 0x00 — Set the color level of Blue to 0, or off.
  - 0x05 — Set the low brightness level to 5, which is 10% in this design.
  - 0x2D — Set the high brightness level to 45, which is 90% in this design.
  - 0x07, 0xD0 — Set the time for one complete pulse to 0x07D0 ms (2000 ms)
  - The command sets the Red level to 100%, Green to 50%, and Blue to 0%, which creates an orange additive color. Then pulse the LED between 10% and 90% brightness with a complete pulse taking 2000 ms.

### 3.6 Effects of Temperature Drift

Temperature drift impacts two portions of this design: the clock system on the MSP430, and the drain current on the P-FET transistors.

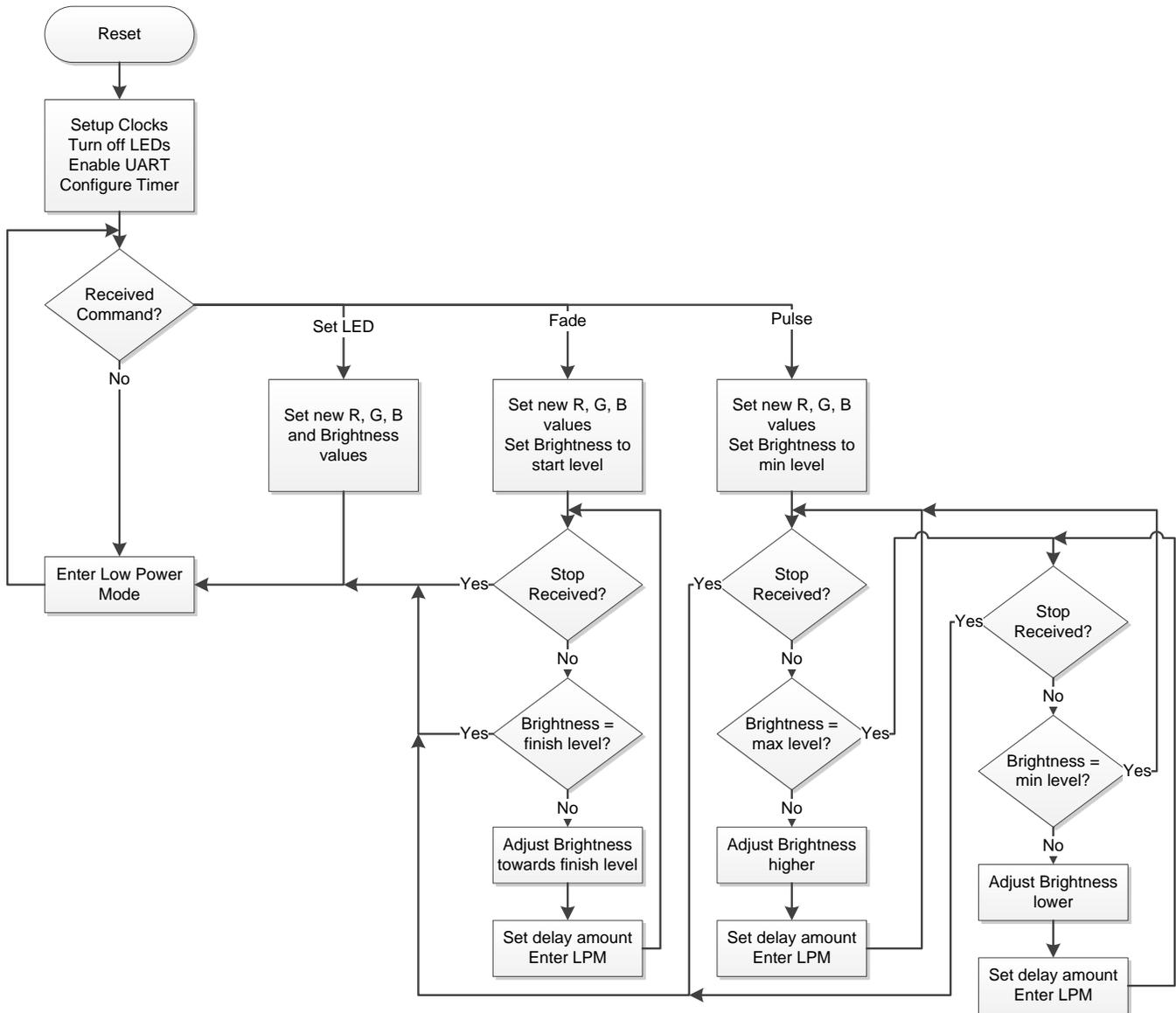
In the MSP430, a change in temperature changes the output frequency of the DCO, which ultimately sources the timer module used to create the three PWMs. However, because the PWMs are all sourced from the same DCO module, any change in DCO frequency will impact all three PWMs the same. This means the PWMs may slightly change in frequency as the DCO varies, but the duty cycle will always remain the same. Because the duty cycle is stable across temperature, there is no variation in color output due to DCO variation.

In the P-FET transistors, an increase in temperature results in a slight increase in drain current. The increase is most noticeable during the triode range, when the gate-to-source voltage ( $V_{GS}$ ) is between the threshold voltage ( $V_T$ ) and 0 V. In the saturation range, when  $V_{GS}$  surpasses  $V_T$ , the current also increases but to a lesser extent. Because logic-level P-FETs are used in this design and the PWM varies between VCC and GND, the P-FETs spend minimal time in the triode range and primarily stay in the saturation and off ranges. This results in a slight increase in current as temperature increases and a decrease as temperature decreases. An increase in current causes the brightness of the LED to increase.

Ultimately, as temperature increases, the brightness of the RGB LED will increase slightly, and as temperature decreases, the brightness will decrease slightly. Using FETs with a more stable temperature coefficient will decrease the level of brightness change.

### 3.7 Software

The main application software flow is shown in [Figure 6](#). The interrupt service routines (ISRs) are shown in [Figure 7](#). The entire software flow is then described in further detail.



**Figure 6. Main Application**

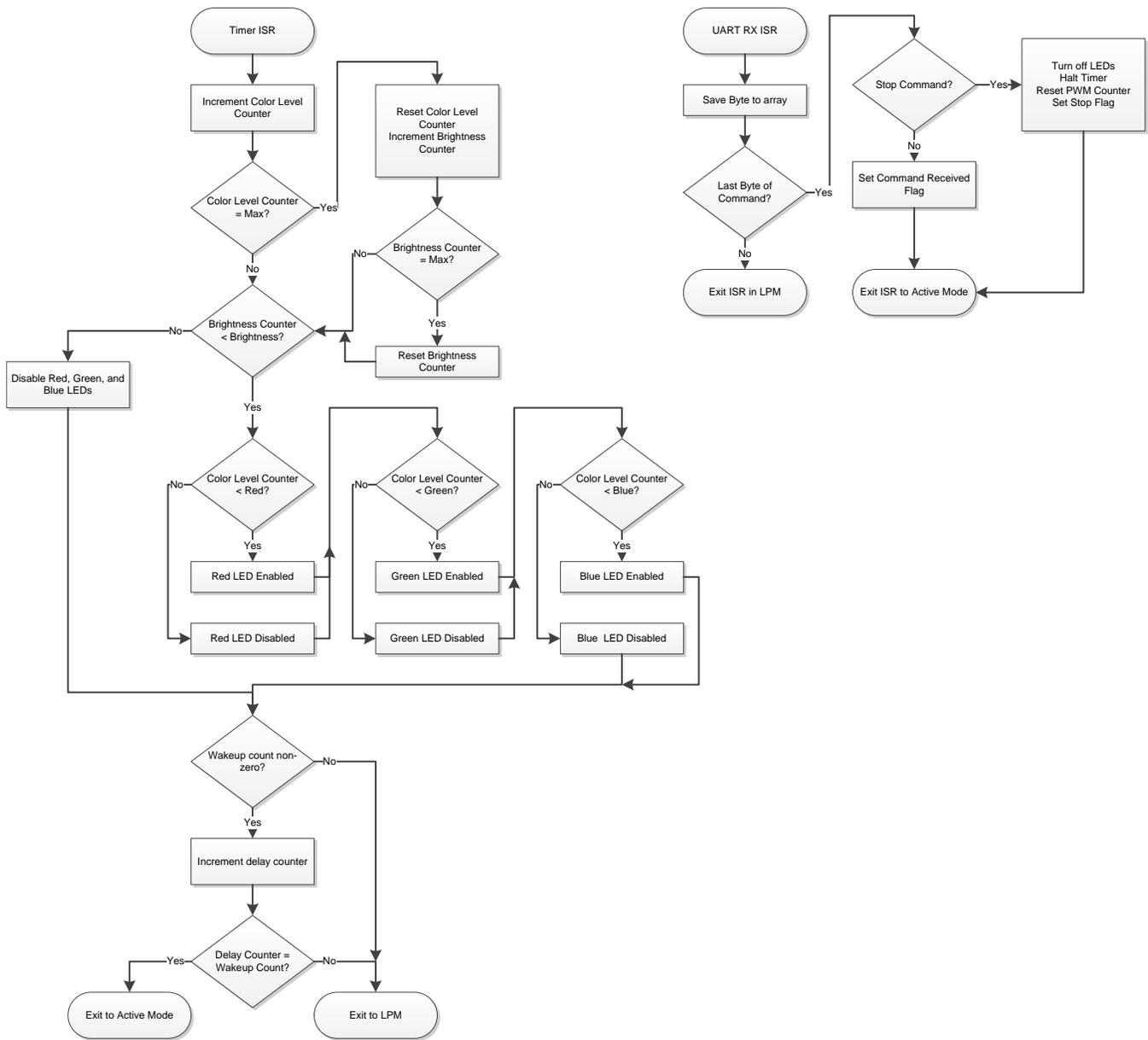


Figure 7. Interrupt Service Routines

### 3.7.1 Main Application

In the initialization routine, all the MSP430 pins are configured and unused pins are configured for the lowest power consumption. Next, the calibrated 8-MHz DCO values are loaded to the DCO control registers. The USCI module is configured for UART mode at 9600 8-N-1 mode. Timer\_A0 is configured to give an interrupt every 200 clock cycles (25  $\mu$ s at 8 MHz) but kept in Stop mode. The maximum value for the color level counter is set to 8 and the maximum for the brightness counter is set to 50. This provides a 9-bit color resolution (three bits per color), which is 512 unique colors. The overall refresh rate of the LED PWM signals is 100 Hz ( $1 / (25 \mu\text{s} \times 8 \times 50) = 1/10 \text{ ms}$ ).

The main loop checks to see if a complete command has been received. If there is no command, the device enters low power mode. If a complete command was received, the command is executed as described below. All commands start with resetting the RGB color levels, both the color level count and the brightness level count, and the wake-up counter. The LEDs are also disabled, and the timer is reset.

#### Set LED

For the Set LED command, the new values for the RGB and brightness levels are set. Then, the device enters low power mode. The timer interrupt will generate the correct waveforms to output the new color.

#### Fade LED

For the Fade LED command, the new levels for RGB are set, along with the initial brightness level. The wake-up count is calculated to determine how long to wait before adjusting the brightness to the next level as part of the effect. The MSP430 enters low power mode to allow the timer interrupt to draw the color at the current brightness level. After the determined number of timer interrupts has passed, the timer ISR exits to active mode. The brightness level is adjusted towards the final level, and the MSP430 is put back into low power mode to repeat the cycle. When the brightness reaches the final level, the Fade subroutine exits and returns to the main loop, re-entering low power mode to await the next serial command. The RGB LED stays configured at the final brightness level.

#### Pulse LED

For the Pulse command, the new levels for RGB are set, along with the initial brightness level, which is the lower brightness of the two levels specified in the command. The wake-up count is calculated to determine how long to wait before increasing the brightness to the next level as part of the effect. The MSP430 enters low power mode to allow the timer interrupt to draw the color at the appropriate brightness level. After the determined number of timer interrupts has passed, the timer ISR exits to active mode. The brightness level is adjusted towards the finish level, and the MSP430 is put back into low power mode to repeat the cycle. When the brightness reaches the highest level, the process repeats, decreasing the brightness level at each step. Once the initial brightness level is reached, the entire procedure starts over to create a continuous pulsing of the LED brightness.

### 3.8 *Timer ISR*

The timer interrupt service routine creates the three color PWMs. First, the color level counter is incremented. If the max color level of 8 is reached, the color level counter is reset to 0 and the brightness counter is incremented. If the max brightness level of 50 is reached, the brightness counter is also reset to 0. The brightness counter is compared to the global brightness level variable. If the counter is higher than the global variable, all three PWMs are set to disable the LED, output high in this design. Otherwise, the color level counter is compared to each color's global variable for level. If the global variable is higher than the color level counter, that color's PWM is set to enable the LED, output low in this design. If the color level counter is higher than the global variable, the LED is disabled, output high in this design.

After all three PWMs have been updated based on the color level and brightness counters, if the wake-up count is active (non-zero), the delay counter is incremented. If the delay counter does not match the specified wake-up count, or the wake-up count is non-active, the ISR exits back to low power mode. If the delay counter matches the wake-up count, the ISR exits to active mode, returning to the main loop.

### 3.9 *UART ISR*

The UART RX ISR saves each incoming byte to an array. The first byte in the command packet is the number of bytes in the command. This command length byte is used to determine when the entire command has been received. If the command is a Stop command, the ISR disables the LEDs by setting the three PWMs to output high, halts the timer, resets the counters, and sets the global stop flag. The ISR then exits to active mode where any active effect will see the stop flag and exit back to the main loop waiting for a new command. If the command is not a Stop command, the ISR sets the global command received flag and exits to active mode. The main loop then executes the command.

## 4 Getting Started: Hardware

### 4.1 Picking an RGB LED and Current Limiting Resistors

This reference design uses an RGB LED with a common cathode. The anodes of the three LEDs are connected to VCC through a current limiting resistor and a P-FET for controlling each color. Any RGB LED can be used, but if N-FETs are used to control each color, the PWM signals should be inverted from what is implemented in this design. Also, some RGB LEDs have diffusers built into the packaging and some do not. Use a diffuser, either built into the LED or added externally, to fully blend the three colors into the resulting additive color.

The current limiting resistors should be picked for each color in the RGB LED so that all three color output levels are balanced. Failure to balance the colors will result in incorrect additive colors on the LED during operation. The datasheet for the RGB LED used provides a current-versus-brightness graph for each color. Use current limiting resistors to set the current for each color that result in comparable brightness. To verify that the resistors picked are correct, set the design to output white (RGB levels all set to the same value) and verify the additive color created by the LED is white.

### 4.2 Obtaining a PCB

As RGB LEDs vary in size, footprint, and configuration, the hardware for this design is not available to order. The design files for the schematic and PCB are available at <http://www.ti.com/tool/TIDM-G2XXSWRGBLED> and can be modified to create the correct circuit for the RGB LED used. Alternatively, because the circuit for controlling an RGB LED is very basic, a prototype can be easily built using a breadboard or similar method.

This design also includes the required Gerber files and BOM to build the exact PCB shown in the guide.

### 4.3 Connecting to the LaunchPad

TI LaunchPads are low cost, easy to use, development kits with common header connections to connect BoosterPacks. BoosterPacks are plug-in modules that connect to LaunchPads to enable additional applications. This design follows the 40-pin BoosterPack pinout standard as shown on the LaunchPad *Build Your Own BoosterPack* page to enable compatibility with the MSP-EXP430G2 LaunchPad. The MSP-EXP430G2 LaunchPad is available to purchase at the TI Store.

The procedure for connecting the design to the LaunchPad for both the BoosterPack and breadboard versions are shown in the following sections.

#### 4.3.1 BoosterPack Version

The RGB LED BoosterPack described in this design was tested with the MSP-EXP430G2 Rev 1.5 LaunchPad. Other LaunchPad revisions might contain differences and should be compared to Rev 1.5 to determine if any changes will cause problems. The *MSP-EXP430G2 User's Guide* contains revision information that can be used to help reconcile these changes as necessary. [1]

When plugging the RGB BoosterPack into the MSP-EXP430G2 LaunchPad, ensure the 3V3 pin on the BoosterPack header aligns with the VCC pin on the LaunchPad and the GND pin on the BoosterPack with the top right GND pin on the LaunchPad.

There are two source pins for each color PWM on the BoosterPack to allow different GPIOs on the MSP430 to source the PWMs depending on the requirements of the design. On the PCB, there is a pair of 0-Ω resistors for each PWM. One 0-Ω resistor should be populated to select the source pin for the PWM, and the other should be left non-populated. The primary PWM pins are on the outer rows of the 40-pin BoosterPack connector and the alternate pins (marked with an \* on the PCB) are on the inner rows of the BoosterPack connector.

The RGB LED BoosterPack uses the following pins on the MSP-EXP430G2 LaunchPad:

- P1.1 — UART RX: Receive pin for serial communication
- P2.0 — Red PWM: PWM output from MSP430 for controlling the red LED
- P2.1 — Green PWM: PWM output from MSP430 for controlling the green LED
- P2.2 — Blue PWM: PWM output from MSP430 for controlling the blue LED

#### 4.3.2 Breadboard Prototype Version

Building the RGB LED circuit without the BoosterPack is simple to do with a bread board or similar prototyping mechanism. The LED driving circuit shown in this design (or similar, depending on the RGB LED used) should be built on the breadboard. Jumper wires can be used to connect VCC, Ground, and the three PWMs (red, green, and blue) from the LaunchPad to the breadboard. The default pins for RGB are P2.0, P2.1, and P2.2, respectively.

### 4.4 Connecting Serial Interface

The UART interface to the MSP430 uses P1.1, configured as UCA0RXD, UART Receiver. The UART transmit pin is not used in this design. The back channel UART in the LaunchPad provides a simple mechanism for communicating with this design through UART. To use the back channel UART, connect the UART jumpers on J3 of the LaunchPad in the hardware mode (see *MSP-EXP430G2 User's Guide* for more information [1]). Only the top jumper is required for MSP430 RX. To use another UART device to send commands to the MSP430, disconnect the UART jumpers on the LaunchPad and connect the UART host transmit pin to the RXD pin on the BoosterPack.

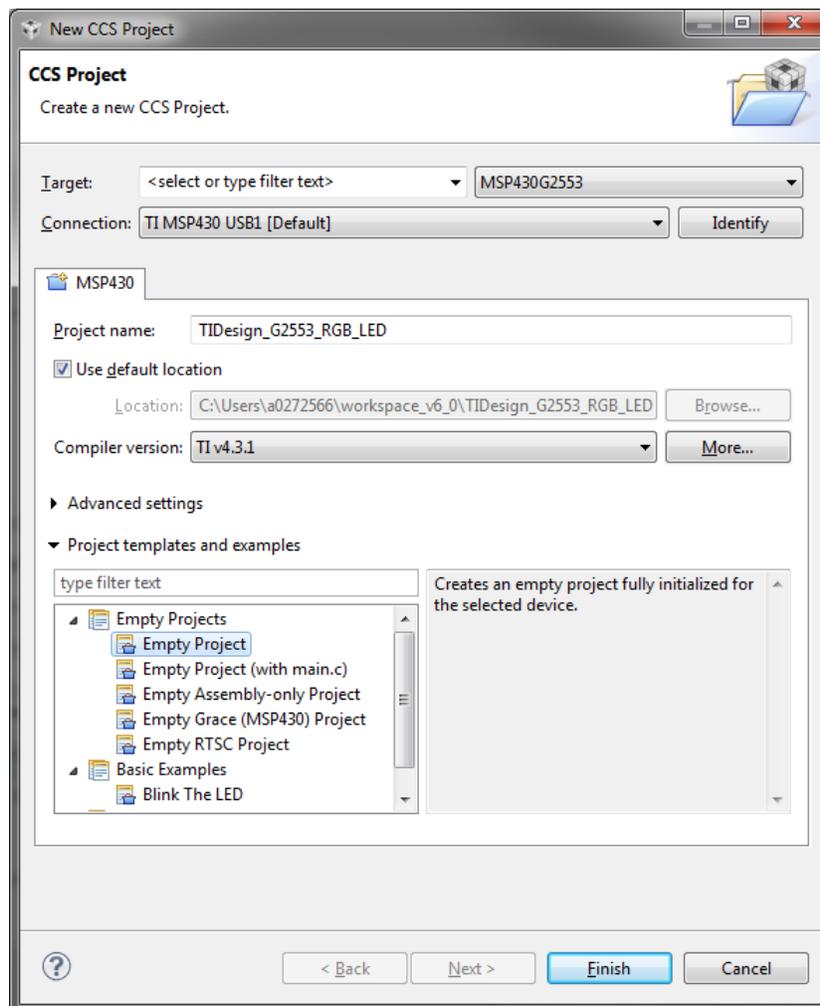
## 5 Getting Started: Firmware

The project included in the software package has been built and tested with CCS 6.0.0.00190 using compiler version TI v4.3.1. The download package includes a single code file, G2xx\_RGB\_LED.c, which should be included into a new CCS project for building.

The procedure to build the project for CCS is described in the following section.

### 5.1 Building Using CCS v6

1. Create a new, blank project (*File*→*New*→*CCS Project*).
2. Set the Target as "MSP430G2553", name the project (for example, "TIDesign\_G2553\_RGB\_LED"), select *Empty Project*, and click *Finish*. See [Figure 8](#) for New CCS Project parameters.



**Figure 8. New Project Settings**

3. Right click on the project name in the *Project Explorer* window, and select *Add Files...*
4. Browse to the location where G2xx\_RGB\_LED.c was saved from the download. Select and click *Open*.
5. On the popup window that appears, select *Copy Files* to copy the file into the project, then click *OK*.
6. Build the project (Ctrl+B, Menu→Project→Build All, or )
7. Connect the RGB LED BoosterPack to the LaunchPad as shown in [Section 4.3](#) and connect the LaunchPad to the PC with the USB cable.

8. Download project to device (F11, Menu→Run→Debug, or )
9. Execute the program () or close debugger and reset device.

## 5.2 RGB LED Demo Mode

This design also includes a demo mode in the firmware to demonstrate the RGB LED control without the need for a serial command. To run the demo mode version of the software, the **#define** DEMO\_MODE line of code at the beginning of the G2xx\_RGB\_LED.c file should be uncommented. Then rebuild, download, and execute the project as shown in Steps 6 through 9 in [Section 5.1](#).

## 5.3 PC Serial Interface through LaunchPad Back Channel UART

A PC serial terminal application can be used to send commands to the design using the LaunchPad's back channel UART. To match the command protocol for this design, it is important to use a serial terminal that allows sending raw hex values instead of ASCII characters.

## 6 Customizing the RGB LED

### 6.1 RGB LED

As mentioned in [Section 4.1](#), the RGB LED in the design can be swapped out for a different model. Because there could be differences in footprint or configuration, the circuitry for the drive circuit may need to be changed as well. If N-FETs are used instead of P-FETs to enable the three-color LEDs, the output of the PWMs should be inverted as well.

### 6.2 Changing GPIOs Used for PWMs

Any GPIO can be used to generate the three PWMs for red, green, and blue. The timer ISR should be modified to use the appropriate GPIOs for generating the PWM. Using the same GPIO port for all three PWMs and using sequential bits in the same port (that is, P2.0, P2.1, and P2.2) will reduce the instruction count for the software required to generate the PWM. Also, using the lower bits (P2.0, P2.1, and P2.2) will be more efficient than the higher bits (P2.7, P2.6, and P2.5). The software takes advantage of comparisons in C returning 0x00 or 0x01 with fewer shift instructions required to put the correct value in the correct bit of the port.

## 6.3 Application

### 6.3.1 MCU System Initialization

The software included in this reference design initializes the MCU system and peripherals as follows:

- Watchdog disabled
- MCLK = SMCLK = DCO = 8 MHz from calibration constants in info memory
- ACLK = unused
- Unused GPIOs are set to output low

### 6.3.2 Changing RGB LED PWM Configuration

The RGB LED PWMs must have an overall refresh rate greater than 50 Hz or flickering will be observable. The eight color levels and 50 brightness levels in this design were used with a 25- $\mu$ s step generated by the Timer\_A0 module to create a 100-Hz overall refresh rate. Both the color level and brightness levels can be adjusted, as well as the timer step size to create custom color and brightness resolutions to match the specific system requirements. For example, if more color resolution but less brightness resolution is needed, a possible configuration would be 64 color levels, 10 brightness levels, and a 25- $\mu$ s step size, resulting in a refresh rate of 62.5 Hz. The limit to how small the time steps can be is determined by the number of clock cycles required to execute the timer ISR to create the PWMs.

### 6.3.3 Adding Effects

The MSP430 can implement light effects using the RGB LED, such as the Fade and Pulse effects already implemented. Additional effects can be added by creating new functions to perform the new effects. These functions should utilize the delay counter, counting the number of timer ISRs, as the timing mechanism for creating the required timings of the effects.

Each effect added that uses the delay timer should also enable the Stop command to exit the effect function. The effect loop should check the status of the exitRoutine flag, which is set when a Stop command is received, and correctly exit out of the effect function, returning to the main loop to wait for a new command.

### 6.3.4 Adding Serial Commands

As new effects are added, the MSP430 should also support additional serial commands to enable using these effects. The serial command protocol described in [Section 3.5](#) allows for a total of 256 unique commands, including the four commands already in the example, and should be used for adding additional commands.

In the `while(1)` main loop of the main function, there is a switch statement, `switch (uartRxBuffer[1])`, that determines which serial command has been received and starts executing the appropriate effect. Adding handling for additional commands is done by creating new cases in the switch statement for the new command. As described in [Section 6.3.3](#), new effects should be implemented using new functions. The new case state in the main loop should only call the function for the new effect. This allows the implemented Stop command to execute correctly.

## 7 Test Data

### 7.1 Power Consumption

There are three operation modes for the MSP430 in this design:

1. LED inactive, waiting for serial command
2. LED active and enabled
3. LED active and disabled

In the two modes during which the LED is active, the time the MSP430 spends in active mode changes depending on the brightness count versus the brightness level. When the brightness count is greater than the brightness level, all LEDs are disabled, which is handled in significantly fewer instructions, resulting in more time spent in low power mode and, therefore, less average current.

All current data shown does not account for the current of the LED, which will vary significantly based on which RGB LED is used and the drive circuitry for the LED.

To get a complete current profile for a design, consider the percentage of time the design spends in each of the three modes. To verify the calculated power profile, disconnect the VCC jumper, J3 on the MSP-EXP430G2 LaunchPad and connect an ammeter in series. To remove the impact of the RGB LED drive circuitry on the measurement, remove the BoosterPack before measuring the current.

#### 7.1.1 Power Consumption — LED Inactive, Waiting for Command

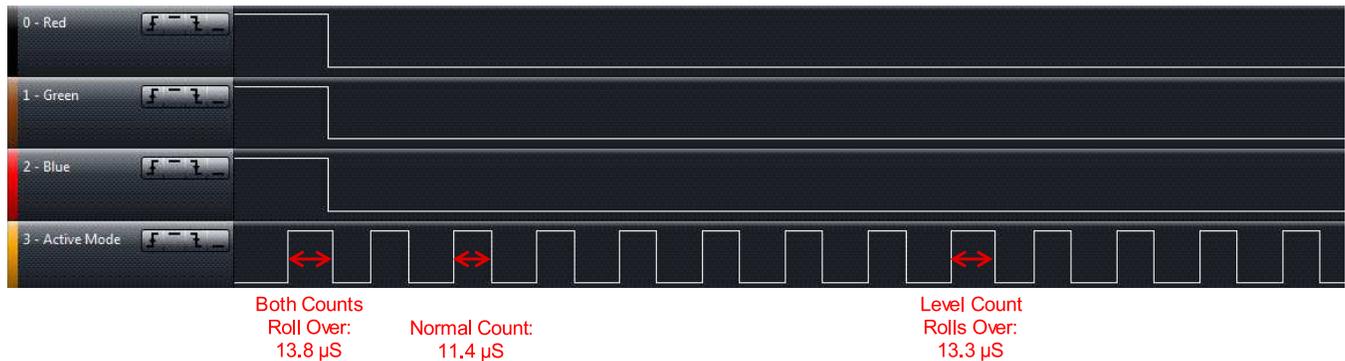
When the RGB LED is inactive, the timer interrupts are disabled and the MSP430 remains in LPM0 without waking up, waiting for UART data on the USCI module. The MSP430 must stay in LPM0 to keep the clock system powered because the UART uses SMCLK to generate a stable baud rate clock. The total current is the datasheet value for LPM0, 56  $\mu$ A at 2.2 V.

### 7.1.2 Power Consumption – LED Active and Enabled

When the RGB LED is active and enabled, the timer interrupt service routine is the only time the MSP430 spends in active mode. Depending on the state of the level and brightness counters, the ISR has three different lengths:

1. Level count increments
2. Level count rolls over, brightness count increments
3. Level count rolls over, brightness count rolls over

Figure 9 shows all three possible lengths, with the measurements for each. In the current design, the normal level count increments seven times, then the level count rolls over and the brightness count increments. This repeats a total of 49 times and on the 50<sup>th</sup> iteration, both the level count and the brightness count roll over, then the entire process repeats.



**Figure 9. Active Mode Timings for LED Active and Enabled**

Table 1 shows the average current consumption during this process, using the datasheet values of 56  $\mu\text{A}$  for LPM0 and 1.3 mA for active mode current at  $f_{\text{DCO}} = 8 \text{ MHz}$ . The timer interrupt that wakes the MSP430 into active mode occurs every 25  $\mu\text{s}$ , giving a total process time of 10 ms ( $25 \mu\text{s} \times 8 \times 50$ ). The LPM0 time is calculated by subtracting the total time in active mode from 10 ms.

**Table 1. Power Budget Average**

FUNCTION	SINGLE DURATION	NUMBER OF OCCURRENCES	TOTAL DURATION	CURRENT	NORMALIZED CURRENT
MSP430 active (normal count)	11.4 $\mu\text{s}$	350	3.9900 ms	1.3 mA	0.648 mA
MSP430 active (level count rolls)	13.3 $\mu\text{s}$	49	0.6517 ms	1.3 mA	0.106 mA
MSP430 active (both counts roll)	13.8 $\mu\text{s}$	1	0.0138 ms	1.3 mA	2.24 $\mu\text{A}$
MSP430 LPM0			5.3445 ms	56 $\mu\text{A}$	37.4 $\mu\text{A}$
<b>TOTAL</b>					0.794 mA

### 7.1.3 Power Consumption — LED Active and Disabled

When the RGB LED is active and disabled, the timer interrupt service routine is the only time the MSP430 spends in active mode. Depending on the state of the level and brightness counters, the ISR has three different lengths:

1. Level count increments
2. Level count rolls over, brightness count increments
3. Level count rolls over, brightness count rolls over

Figure 10 shows all three possible lengths, with the measurements for each. In the current design, the normal level count increments seven times, then the level count rolls and the brightness count increments. This repeats a total of 49 times and on the 50<sup>th</sup> iteration. Both the level count and the brightness count roll over. Then, the entire process repeats.

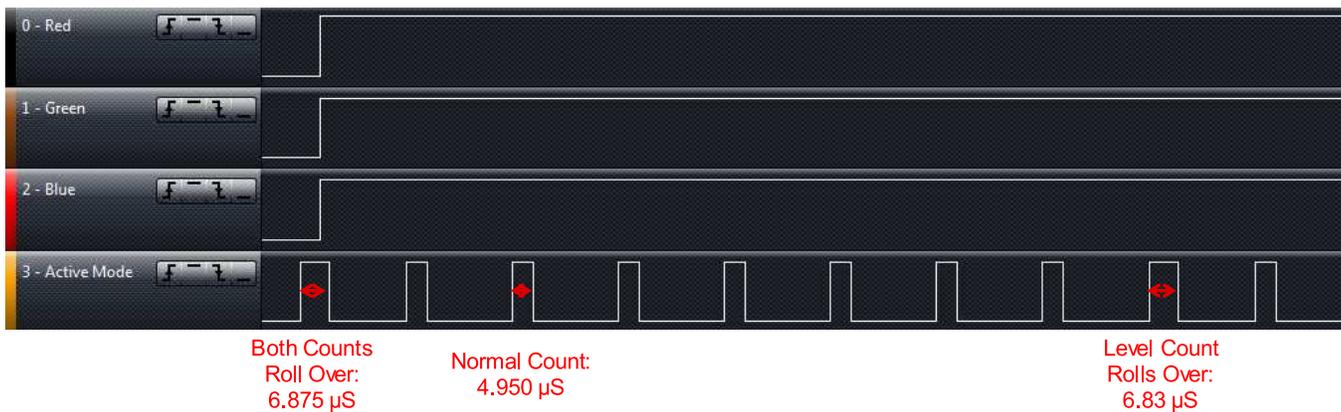


Figure 10. Active Mode Timings for LED Active and Enabled

Table 2 shows the average current consumption during this process, using the datasheet values of 56  $\mu\text{A}$  for LPM0 and 1.3 mA for active mode current at  $f_{\text{DCO}} = 8 \text{ MHz}$ . The timer interrupt that wakes the MSP430 into active mode occurs every 25  $\mu\text{s}$ , giving a total process time of 10 ms ( $25 \mu\text{s} \times 8 \times 50$ ). The LPM0 time is calculated by subtracting the total time in active mode from 10 ms.

Table 2. Power Budget Average

FUNCTION	SINGLE DURATION	NUMBER OF OCCURRENCES	TOTAL DURATION	CURRENT	NORMALIZED CURRENT
MSP430 Active (Normal Count)	4.95 $\mu\text{s}$	350	1.7325 ms	1.3 mA	0.282 mA
MSP430 Active (Level Count Rolls)	6.83 $\mu\text{s}$	49	0.3347 ms	1.3 mA	54.4 $\mu\text{A}$
MSP430 Active (Both Counts Roll)	6.87 $\mu\text{s}$	1	0.0068 ms	1.3 mA	1.11 $\mu\text{A}$
MSP430 LPM0			5.3445 ms	56 $\mu\text{A}$	37.4 $\mu\text{A}$
<b>TOTAL</b>	0.375 mA				

## 8 Design Files

### 8.1 Schematics

To download the most recent schematics, see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.2 Bill of Materials

To download the most recent bill of materials (BOM), see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.3 Layer Plots

To download the most recent layer plots, see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.4 Altium Project

To download the most recent Altium project files, see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.5 Gerber Files

To download the most recent Gerber files, see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.6 Assembly Drawings

To download the most recent assembly drawings, see the design files at [TIDM-G2XXSWRGBLED](#).

### 8.7 Software Files

To download the most recent software files, see the design files at [TIDM-G2XXSWRGBLED](#).

## 9 References

1. Texas Instruments, *MSP430x2xx Family User's Guide* ([SLAU144](#))
2. Texas Instruments, *MSP-EXP430G2 LaunchPad Evaluation Kit User's Guide* ([SLAU318](#))
3. Texas Instruments, *Build Your Own Boosterpack* ([www.ti.com/byob](http://www.ti.com/byob))

## 10 About the Author

**MIKE PRIDGEN** is an applications engineer on the MSP430 Customer Applications Team at Texas Instruments. Pridgen joined TI in 2012 into the Digital LEAP Program where he rotated through various business groups and expand his knowledge across several TI products. He has focused on the MSP430 ultra-low-power microcontroller family with specialization in direct customer support for MSP430G2xx devices.

## IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers ("Buyers") who are developing systems that incorporate TI semiconductor products (also referred to herein as "components"). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer's systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to 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.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER'S USE OF TI REFERENCE DESIGNS.

TI reserves 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 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 for TI components 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.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs 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.

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 that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures 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 Buyer's 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 an agreement specifically governing such use.

Only those TI components that 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 that have **not** been so designated is solely at Buyer's risk, and 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.