UART Software Controlled RGB LED Color Mixing With MSP430™ MCUs

Introduction

Red, green, and blue (RGB) light emitting diodes (LEDs) are used in many applications such as user interfaces and lighting. These LEDs work on the principle of color mixing by varying the relative intensity of the red, green, and blue LED to produce different colors. Color-mixing using a microcontroller (MCU) is achieved by controlling the LED with PWM (pulse width modulated) signals, where the frequency stays above approximately 60 Hz to prevent flicker visible to the human eye. By varying the duty cycle of the red, green, and blue LED different colors can be achieved. The implementation presented here is a UART-controlled RGB color mixing solution. It has been optimized for lowest code size, fitting in a low-cost 0.5KB MSP430FR2000 MCU, and with limited timer resources (one Timer B with 3 CCR capture compare registers), yet still provides 12 different color options, selectable with a UART command. To get started, download project files and a code example demonstrating this functionality.

Implementation

The solution uses an MSP430FR2000 MCU and an external RGB LED with a current limiting resistor and P-FET for each of the three LED colors. This allows the LED to be controlled by the MSP430™ MCU, but driven by more current than should be supplied directly from an MSP430 device output pin. The LED BoosterPack™ plug-in module used comes from the TIDM-G2XXSWRGBLED TI Design – schematics and additional information can be found there. The MSP430FR2000 MCU was used with the MSP-TS430PW20 target socket board and connected with wires to the BoosterPack module as shown in Figure 1. The backchannel UART of the MSP-FET programmer and debugger or eZ-FET on an MSP430™ LaunchPad™ development kit was used for the UART communication with a terminal program on the PC to send the commands for selecting each color.

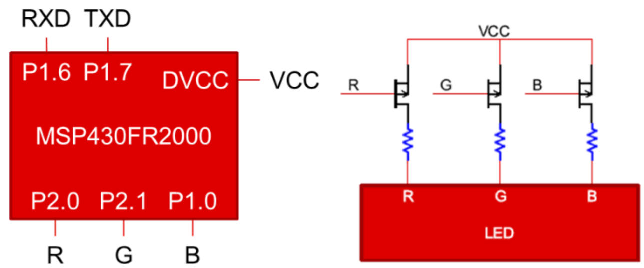

Figure 1. RGB LED Controller Block Diagram

The theory of RGB color mixing using PWMs is further elaborated on in the *MSP430 Software RGB LED Control Design Guide* Section 3.1, *RGB Color Mixing*, and Section 3.2, *LED Control*. However, to save code space a different methodology is used to create the PWMs. The PWMs are controlled by the TB0CCR0 (Blue), TB0CCR1 (Red), and TB0CCR2 (Green) registers in the Timer B0 module. Typically, when generating PWMs with the timer module the timer is used in Up Mode and TB0CCR0 sets the period. With that method only (Number of CCRs – 1) PWMs can be generated on a single timer module – with the Timer B0 with 3 CCRs on the MSP430FR2000 device, only 2 PWMs would be possible. Therefore, a different method is used to create the PWMs using continuous mode. The theory behind this hardware timer plus software ISR handling approach is explained further in *Multiple Time Bases on a Single MSP430 Timer Module*.

Because the frequency only needs to be approximately 60 Hz to prevent visible flicker, the master clock (MCLK) can remain at the default 1.048 MHz without running into the limitations presented in *Multiple Time Bases on a Single MSP430 Timer Module*. The PWMs are generated from the auxiliary clock (ACLK) sourced from the internal trimmed low-frequency reference oscillator (REFO) at 32768 Hz. The clock is divided by 4 to produce 8192 Hz as the timer clock – this makes it so that the period of 60 Hz can be generated with TB0CCRx values totaling 135 for the period value. Because 135 is less than 255 (FFh) the lookup tables for the timer periods for different RGB values can be made up of 8-bit values, saving additional program memory space on small devices. Note that the LEDs are turned on during the low phase of the PWM, so colorsLow[] contains the values for the LED on period and colorsHigh[] contains the values for the LED off period.

Performance

To run the demo, connect the hardware as previously described, load the code into the device, allow the device to run and end the debug session. Note that the MSP-TS430PW20 target board already includes the correct connections for the UART TXD and RXD on the MSP-FET connector as long as JP14 and JP15 are populated (leave JP13 unconnected). At startup, the LED will appear white as the device initialization occurs (because the P-FETs controlling the LED are active low). Once the initialization is complete the LED will default to red. Using the backchannel UART on the MSP-FET or the eZ-FET, use a terminal program on the PC set to 9600 baud none parity 1 stop bit to select the colors. There are 12 colors, selectable by sending a single hex byte of 0 to Bh. Any invalid value defaults to red.

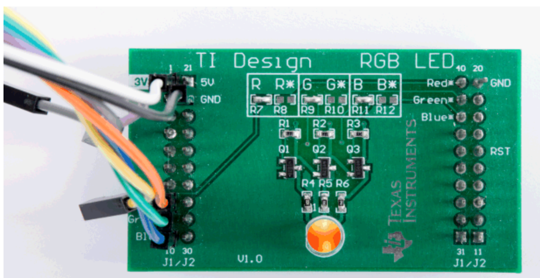

Figure 2. RGB LED Demo

The demo produces 12 colors tuned to the LED on the BoosterPack module. If a larger MSP430 MCU were used, then many more colors could be added. For this RGB LED, the red LED has a lower intensity than the green or blue LEDs. Therefore, the values in colorsHigh[] and colorsLow[] were tuned to have less green and blue when mixing with red to produce the desired colors. For best viewing, use a diffuser so that the three LEDs can fully blend into the resultant color.

This solution provides RGB LED control with minimal external components and optimized software that fits in code-limited devices down to 0.5KB.

Device Recommendations

The device used in this example is part of the MSP430 Value Line Sensing portfolio of low-cost MCUs, designed for sensing and measurement applications. This example can be used with the devices shown in Table 1 with minimal code changes. For more information on the entire Value Line Sensing MCU portfolio, visit www.ti.com/MSP430ValueLine.

Table 1. Device Recommendations

Part Number	Key Features
MSP430FR2000	0.5KB FRAM, 0.5KB RAM, eComp
MSP430FR2100	1KB FRAM, 0.5KB RAM, 10-bit ADC, eComp
MSP430FR2110	2KB FRAM, 1KB of RAM, 10-bit ADC, eComp
MSP430FR2111	3.75KB FRAM, 1KB RAM, 10-bit ADC, eComp

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES