Bluegiga Forums / Community Forums / Bluetooth Smart

PWM output for LED

Answered

Cynthia De Dalmady
asked this on February 20, 2014, 06:16

Share Tweet 0 Like 0

I want to power an RGB LED, but I have a few questions. What ports are capable of PWM output? The documentation describes using the function call hardware_timer_comparator(timer, channel, mode, comparator_value) to generate the output. How does the function work? What does it mean by the channel? How can I use this to make the led blink? I other words how I toggle from sending the signal to turning it off? Would this be the duty cycle?

One person would like this to be answered. Me too!

Comments

Jeff Rowberg
Bluegiga Technologies

Hi Cynthia,

The best starting point for PWM output is to look at these two example projects:

- https://bluegiga.zendesk.com/entries/23508197--BGScript-pwm-4channel-1khz-Fast-4-channel-PWM-output-at-1-KHz
- https://bluegiga.zendesk.com/entries/23498301--BGScript-pwm-4channel-1hz-Slow-4-channel-PWM-output-at-1-Hz

The **hardware_timer_comparator** command is documented in the API reference guide, which you can get from the BLE112 documentation and downloads page under the "User Guides" heading:

- http://www.bluegiga.com/en-US/products/bluetooth-4.0-modules/ble112-bluetooth--smart-module/documentation

The particular channel depends on which timer you are using. Timer 1 has five channels, one for a baseline frequency and four for outputs at various (or similar) duty cycles, as shown in the two examples above. Timers 3 and 4 have two channels. The 1 Hz example using Timer 1 would be the best starting point for an LED to blink visibly, while the 1 kHz example would be better for fast PWM for varying the intensity of RGB LED emitters.

You can turn off the PWM output completely by setting the **comparator_value** argument (the last one in the command) to 0. This is the same value used to set the duty cycle.

February 20, 2014, 21:33

darren delorey

This is a very common BLE peripheral control application and I am surprised that Bluegiga has not yet published example code for it. Jeff, can we expect an application note addressing an RGB control program? (preferably using an android app. to control the peripheral).

April 1, 2014, 16:33

Jon Prevo
Aero Beacons

I have an RGB LED running on timer 1, Alt. 1 which is connected to P0_2 (Green), P0_3 (Blue) & P0_4 (Red). I'm running through a variety of duty cycles for each color but cannot get it to display different colors. I can only do 7 colors as long as I keep the duty cycles equal (or off).

So all 3 set to a comparator_value of 0-32000 at the same value at the same time produces white. But change that value on any color by only 1 and it will either become the primary color or get totally washed out.

I have current limiting resistors in place as well.

Any ideas? I assume I'm missing something very obvious.

Also, setting the comparator_value to 0 does not turn it off. When running in Mode 2 (from hardware.xml), Red only works if it's set to 0. Weird.

May 7, 2014, 02:57

Jeff, having had a new little one arriving in our family as well, I know you're sleep deprived so I'll be as quick as possible;)

We've got all of this worked out except one crucial part. We're using both the 113 and the 121lr as the MCU in our product since it's a very simple design.

Support

Mike Smartt
Auto Flight
Systems, LLC

We've got the IMU part working great but control of the RGB LED has given us a headache ... or two! While we can successfully control the output voltage (brightness) to the LED manually using "***call hardware_timer_comparator(1, 'pin', 6, 16000)***" for example, we're having trouble with regards to how to apply this **programmatically with BGScript** in our UART. We've done this successfully with an external MCU but would really like to eliminate the need for that to save power.

Here's the basic outline:

Mobile app communicating with a 113/121 using UART. It has 3 sliders in the UI to adjust the brightness of the 3 channels of an RGB LED individually. On the BLE113/121 hardware this is P1_1, P1_0 and P0_7.

We can turn them on and off in the connected/disconnected functions but where would you put the logic for control of individual output? And what would that look like?

*event **attributes_value**(connection, reason, handle, offset, value_len, value_data)**?***

*event **attclient_indicated**(connection, attrhandle)**?***

*event **system_endpoint_watermark_rx**(endpoint, size)**?***

We spent 8 hours on this yesterday which is **embarrassing** because we know it's a **very simple** procedure in logic we understand e.g.: C or SWIFT. Could you maybe provide an example script of what this might look like? This is the last hurdle for us to move to the hardware part of production!

THANK YOU!:)

March 19, 2015, 22:25

Jeff Rowberg
Bluegiga
Technologies

Hi Mike, **Answer**

The **hardware_timer_comparator** API command is the correct one to use for adjusting the brightness level (duty cycle) of a PWM output signal. You can use the **attributes_value** event to catch remotely written values from a connected BLE client, and parse whatever bytes are written to decide how to change the PWM settings. I would recommend looking at the server/slave end of the **gpio_remote** example project online:

- https://bluegiga.zendesk.com/entries/98433486--BGScript-gpio-remote-Remote-GPIO-control-with-master-and-slave-components

This uses GATT operations for remotely controlling digital GPIO output levels, but something similar could be used for remote control of PWM with some small modifications.

For UART control, I would probably go with a fixed-size watermark to received known-length packets, which would allow you to more easily parse data instead of reading in one or more bytes at a time manually into a buffer until you have enough for a full command "packet" in whatever protocol you define. For this purpose, I would recommend looking at the following resources online:

- https://bluegiga.zendesk.com/entries/23161557--REFERENCE-Endpoint-watermarking-with-BGScript-BGAPI
- https://bluegiga.zendesk.com/entries/28461493--BGScript-uart-echo-packet-UART1-Alt1-loopback-local-echo-with-watermarking-and-packetization
- https://bluegiga.zendesk.com/entries/29185293--BGScript-spp-over-ble-AT-command-SPP-implementation-for-BLE

The first two should hopefully be pretty self-explanatory in how they relate to your questions, but the 3rd one is more complicated. It implements (in BGScript) an AT-style command parser using RX watermarks and an application-based command buffer. This is possibly more flexible than you need, and also more complicated. But you can see how it is implemented if you look at the **system_endpoint_watermark_rx** event handler.

March 20, 2015, 22:56

Add a comment

Save comment