Control Hardware PWM frequency

Asked 11 years, 4 months ago Modified 4 years, 11 months ago Viewed 108k times

25

I'm using the hardware PWM output with wiringpi. It provides the function pwmSetClock that should make it possible to change the frequency. (https://projects.drogon.net/raspberry-pi/wiringpi/functions/). I believe since the default is 200 Mhz setting the divisor to 200000000 should make an LED hooked up to the output flash visibly, but that is not the case.

Can this be changed?

gpio hardware pwm wiringpi

Share Improve this question Follow

edited Dec 15, 2015 at 20:17
Ghanima ♦ 15.9k 16 62 119

asked Feb 8, 2013 at 14:59
user1217949 767 1 7 14

1 I am doing some testing with the hardware PWM, and it doesn't seem to have a fiexd frequency. It varied based on the pulse width set in `pwmWrite()` . Not something I would expect to happen – TheMeaningfulEngineer Aug 8, 2013 at 13:13

3 Answers

Sorted by: Highest score (default)

31

I've recently had some reason to start experimenting with PWM myself, and found that (as pointed out by one of the comments) the frequency seems to vary with duty cycle - bizzare, right? It turns out that Broadcom implemented "balanced" PWM in order to make the on and off PWM pulses as evenly distributed as possible. They give a description of the algorithm and some more discussion on page 139 of their datasheet: http://www.element14.com/community/servlet/JiveServlet/downloadBody/43016-102-1-231518/Broadcom.Datasheet.pdf

So what you really want is to put PWM in mark-space mode, which will give you the traditional (and easily predictable) PWM you're looking for:

```
pwmSetMode(PWM_MODE_MS);
```

The rest of the answer assumes we're in mark-space mode.

I also did some experimenting with the allowable range of values for `pwmSetClock()` and `pwmSetRange()` . As noted in one of the other answers, the valid range for `pwmSetClock()`

seems to go from 2 to 4095, while the valid range for `pwmSetRange()` is up to 4096 (I didn't attempt to find a lower-limit).

The range and clock (a better name is probably divisor) both affect the frequency. The range also affects resolution, so while it may be possible to use very low values, there is a practical limit to how low you will probably want to go. For example, if you used a range of 4, you could achieve higher frequencies, but you will only be able to set the duty cycle to 0/4, 1/4, 2/4, 3/4 or 4/4.

The Raspberry Pi PWM clock has a base frequency of 19.2 MHz. This frequency, divided by the argument to `pwmSetClock()`, is the frequency at which the PWM counter is incremented. When the counter reaches a value equal to the specified range, it resets to zero. While the counter is less than the specified duty cycle, the output is high, otherwise the output is low.

This means, if you want to set the PWM to have a specific frequency, you can use the following relationship:

```
pwmFrequency in Hz = 19.2e6 Hz / pwmClock / pwmRange.
```

If you use the maximum permissible values for `pwmSetClock()` and `pwmSetRange()`, you will end up with the minimum achievable hardware PWM frequency of ~1.14 Hz. This will certainly give a visible flicker (more of a flash, really) to an LED. I did confirm the above equation with an oscilloscope, and it seems to hold. The upper frequency limit will be affected by the resolution you need, as described above.

Share Improve this answer Follow

edited Jun 9, 2014 at 17:09
Seanny123
105 5

answered Sep 23, 2013 at 13:16
Kerry
411 4 5

Concerning the lower bound on pwmRange: I successfully set it to 2 (to get a duty cycle of 50%). – Ted Pudlik Jul 19, 2014 at 21:32

1 from what source do you know that the pwm clock has a frequency of 19.2 mhz? – thi gg Dec 3, 2014 at 17:37

14

According to this formula:

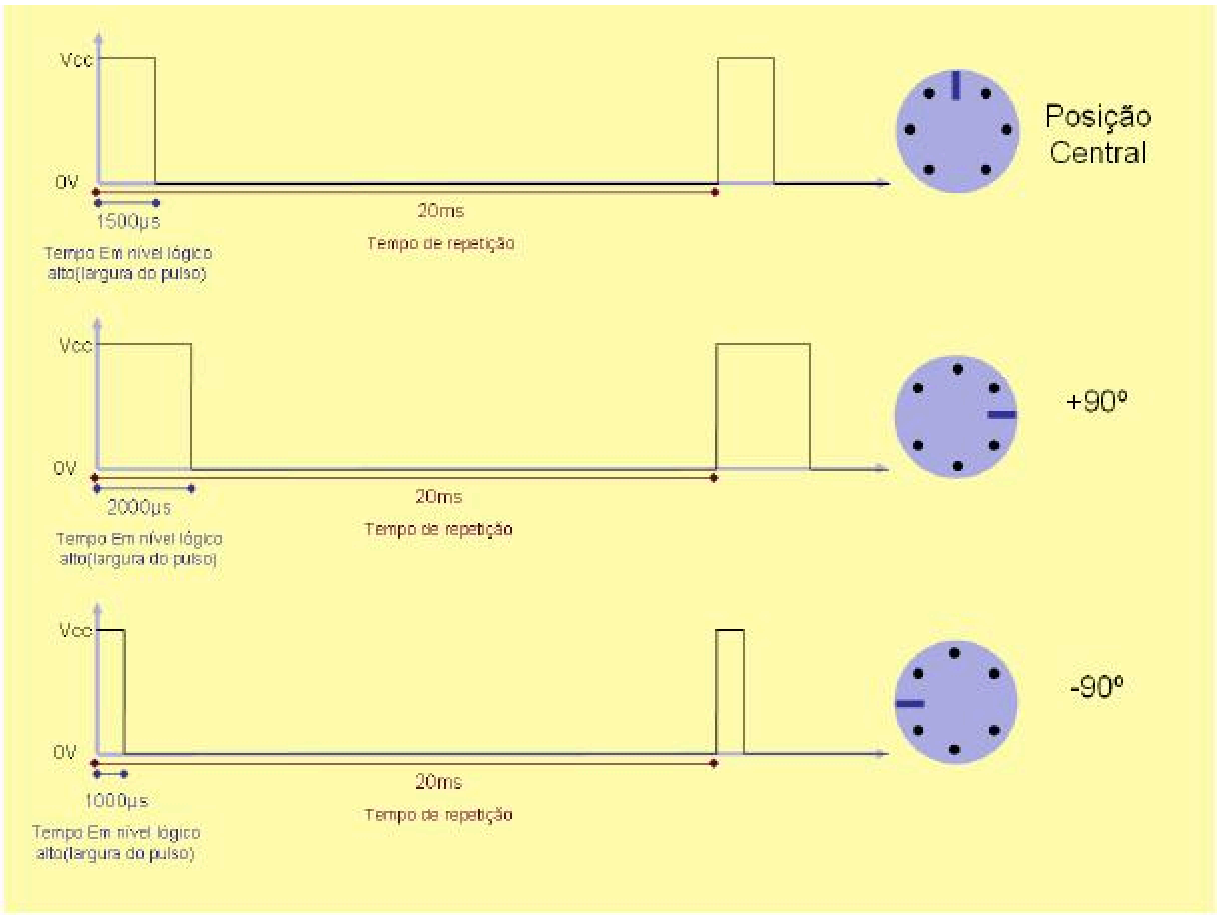
```
pwmFrequency in Hz = 19.2e6 Hz / pwmClock / pwmRange
```

We can set `pwmClock=1920` and `pwmRange=200` to get `pwmFrequency=50Hz`:

```
50 Hz = 19.2e6 Hz / 1920 / 200
```

I test it on alarmpi:

```
$ pacman -S wiringpi
$ gpio mode 1 pwm
$ gpio pwm-ms
$ gpio pwmc 1920
$ gpio pwmr 200     # 0.1 ms per unit
$ gpio pwm 1 15     # 1.5 ms (0°)
$ gpio pwm 1 20     # 2.0 ms (+90°)
$ gpio pwm 1 10     # 1.0 ms (-90°)
```



Vcc
0V
1500µs
Tempo Em nível lógico alto(largura do pulso)
20ms
Tempo de repetição
Posição Central
Vcc
0V
2000µs
Tempo Em nível lógico alto(largura do pulso)
20ms
Tempo de repetição
+90º
Vcc
0V
1000µs
Tempo Em nível lógico alto(largura do pulso)
20ms
Tempo de repetição
-90º

Note: My servo expects a 50Hz signal.

Share Improve this answer Follow edited Nov 11, 2015 at 8:50 answered Nov 11, 2015 at 8:43

kev
356 2 5

how do you come to: 'gpio pwmr 200 # 0.1 ms per unit' – mxlian Feb 5, 2017 at 21:34

50Hz ---> 20ms per cycle. 20ms / 200 units = 0.1ms per unit – mxlian Feb 5, 2017 at 21:40

6

This is the code I'm using. I'm trying to see what will change as i change the settings.

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

int main (void)
{
  printf ("Raspberry Pi wiringPi test program\n") ;

  if (wiringPiSetupGpio() == -1)
    exit (1) ;

  pinMode(18,PWM_OUTPUT);
  pwmSetClock(2);
  pwmSetRange (10) ;
  pwmWrite (18, 5);

for (;;) delay (1000) ;
}
```

pwmSetClock(1);	-> 2.342kHz
pwmSetClock(2);	-> 4.81MHz
pwmSetClock(3);	-> 3.19MHz
pwmSetClock(4);	-> 2.398MHz
pwmSetClock(5);	-> 1.919MHz
pwmSetClock(6);	-> 1.6MHz
pwmSetClock(7);	-> 1.3MHz
pwmSetClock(8);	-> 1.2MHz
pwmSetClock(9);	-> 1.067MHz
pwmSetClock(10);	-> 959kHz
pwmSetClock(11);	-> 871kHz
pwmSetClock(20);	-> 480kHz
pwmSetClock(200);	-> 48kHz
pwmSetClock(500);	-> 19kHz
pwmSetClock(1000);	-> 9.59kHz
pwmSetClock(2000);	-> 4.802kHz
pwmSetClock(4000);	-> 2.401kHz
pwmSetClock(5000);	-> 10.58kHz

From what i've tested, it seems the divisor goes from 2 to some number less than 5000. I would guess it has something to do with the binary representation of those numbers which are being directlly set in the register. Once the numbers binary representation has more bits than the register can take it just takes the first bits and interpretes the numbers that way. Thats why the strange behaviour comes when going from 4000 to 5000.

Share Improve this answer Follow

answered Aug 8, 2013 at 14:00

TheMeaningfulEngineer
279 1 6 12

2 How would I change the duty cycle? – noufal Dec 24, 2013 at 11:29

How did you measure the frequencies? – Seanny123 Jun 9, 2014 at 14:56