Hardware SPI (HSPI) Clock Configuration Registers Explained

esp8266.com/viewtopic.php

Code: Select all
```
#define SPI_CLOCK(i)                                    (REG_SPI_BASE(i)   + 0x18)
#define SPI_CLK_EQU_SYSCLK (BIT(31))
#define SPI_CLKDIV_PRE 0x00001FFF
#define SPI_CLKDIV_PRE_S 18
#define SPI_CLKCNT_N 0x0000003F
#define SPI_CLKCNT_N_S 12
#define SPI_CLKCNT_H 0x0000003F
#define SPI_CLKCNT_H_S 6
#define SPI_CLKCNT_L 0x0000003F
#define SPI_CLKCNT_L_S 0
```

This register range sets the SPI clock frequency (For SPI or HSPI). There are 5 main variables you can set.

- SPI_CLK_EQU_SYSCLK
Setting this bit (and only this bit) will give you a clean SPI clock @ 80MHz (same speed as CPU SYS Clock). Pretty straightforward. If you enable the other bits as well, you might get weird lower clock pulses superimposed on your 80Mhz clock.

To set this, just use:

Code: Select all
```
WRITE_PERI_REG(SPI_CLOCK(HSPI), SPI_CLK_EQU_SYSCLK);
```

You also need to set bit9 in the PERIPHS_IO_MUX register. This is explained in the Espressif spi.c file comments.

You won't normally be touching SPI, so I've used HSPI hardcoded here. If you are modifying the standard SPI.c file, then it'd have spi_no instead of HSPI.

- SPI_CLKDIV_PRE
This is a clock pre-divider. It takes your system clock of 80MHz and divides it down to a lower frequency before it gets passed into the SPI clock circuit. The frequency set here is used for the next section where it defines the length of a CLKCNT (clock count). The value you set here is always one less than the division ratio. That is, the division ratio is N+1. Setting a zero indicates a division of 1. Setting a 7 results in a division of 7+1 = 8 to give 10MHz.

- SPI_CLKCNT_N
This is the number of pre-divided clocks you want to use to define one entire SPI clock pulse. Again, its N+1, so set it to one less than what you want. (Using the pre-division ratio of 8 above) If you set SPI_CLKCNT_N to 9, you would end up with an SPI clock frequency of 1MHz ([80 / 8] / 10 = 1MHz).

- SPI_CLKCNT_H & SPI_CLKCNT_L
These registers set the number of CLKs in each group of CLKCNT_N pulses that the SPI clock is high or low. For a normal SPI clock, you want the High and Low periods to be the same. Following on from the example above, you would want the clock to be high for 5 out of the 10 clock counts, and low for the remaining five.

Now, each register, _H and _L, don't correspond exactly to the number of clocks for each period. If they did, you could set both to 3, and what happens during the remaining 4 clock periods? If you did set both to 3, you would find the SPI clock is low all the time! Why?

Both registers need to be used together to determine the ratio between high and low periods. It takes the difference between the two registers and sets the ratio from there.

If SPI_CLKCNT_H is higher in value than SPI_CLKCNT_L, then the difference is the number of clock periods that the SPI clock is HIGH.

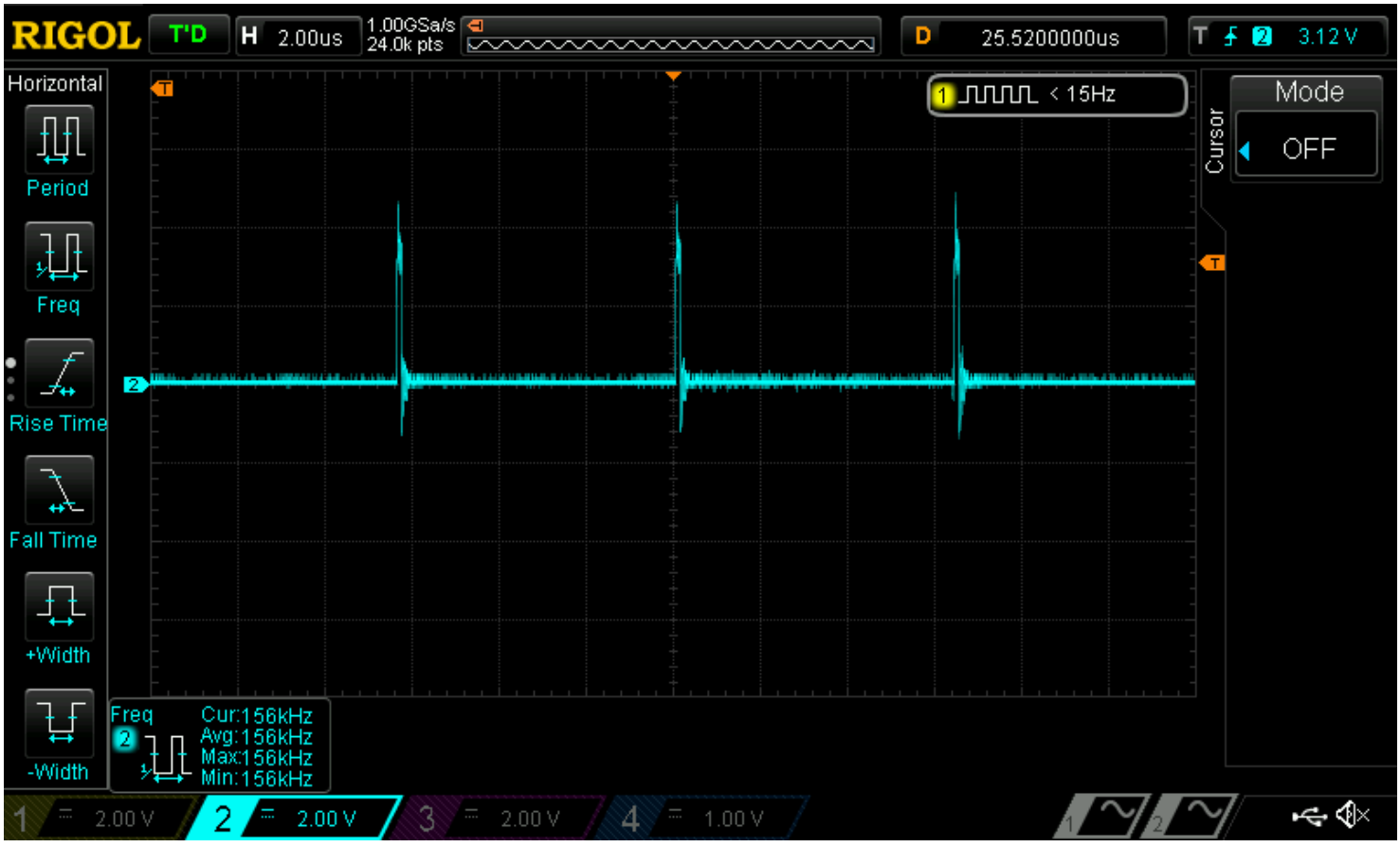
If SPI_CLKCNT_H is lower in value than SPI_CLKCNT_L, then the difference is the number of clock periods that the SPI clock is LOW.

Both SPI_CLKCNT_H and SPI_CLKCNT_L must be lower than SPI_CLKCNT_N. Not sure why, but setting values higher breaks the clock output.

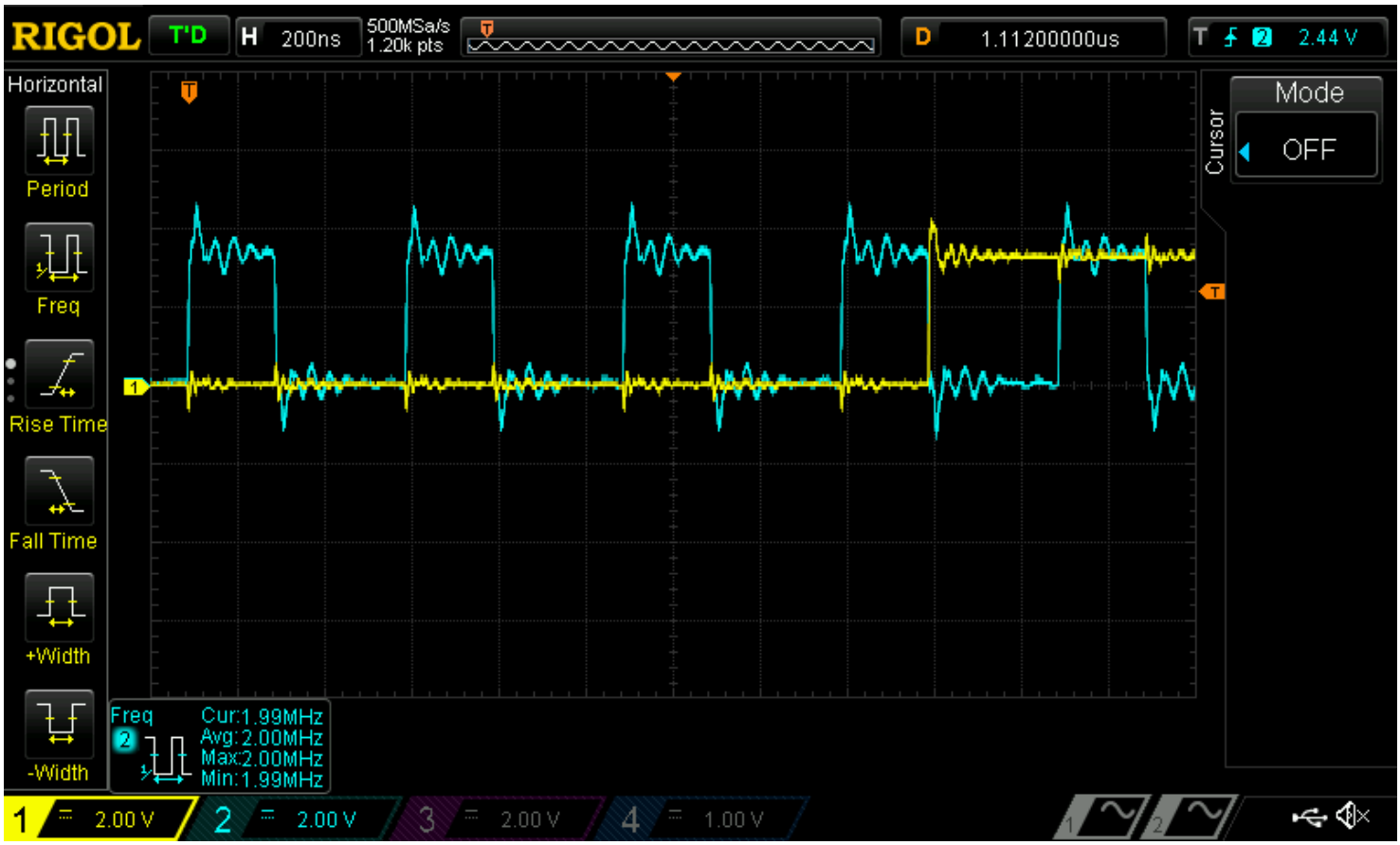
These are my findings by playing around with the various values for each register, and observing the clock output on an oscilloscope. I'm pretty sure this is how it all works, and so far the assumptions above have held for all the values I've tested.

I have some screenshots and examples of values I used to achieve them. I'll upload them soon. If anyone has any further info on the SPI subsystem and what all the registers do, please share  My next challenge is working out how all the transmit/receive buffers work.

This is what you get if you set SPI_CLKCNT_H to be 0x01 and SPI_CLKCNT_L to be 0x00 (difference is 1, _H > _L, so high for 1 clock). I had it set to count 64 clock cycles here. (0x3F & SPI_CLKCNT_N) so the freq was ~156kHz (prediv of /8)



Using a pre-div value of 7 (to divide by 8), and a CLKCNT_N value of 4 (to count 5 clocks) gave me a divide by 40 to get 2MHz which I needed for my SPI EEPROM. I set SPI_CLKCNT_H to 0x04 and SPI_CLKCNT_L to 0x02 resulting in a high period for 2 clocks, and a low period for 3 clocks.



And here is the 80MHz signal on my scope 😀 Breadboard and long probe ground leads make a mess of it, but its still decodable by the scope.