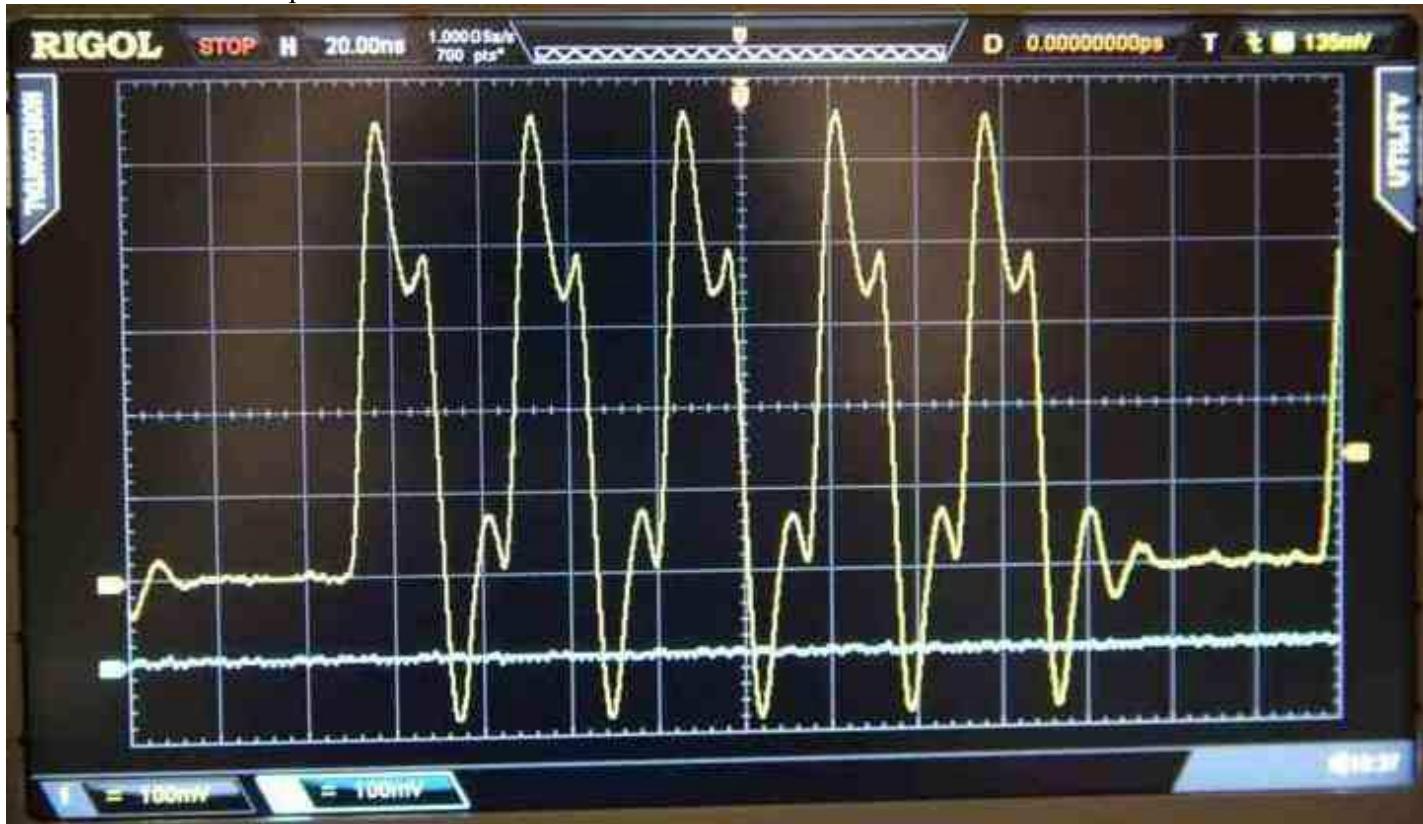


Could somebody help me to determine why my board is running at third of the speed. I am using emBlocks IDE and STM32F429I-Discovery_FW_V1.0.1 library. The loader code calls SystemInit (row 337 in system_stm32F4xx.c), which calls SetSysClock (row 470) executing rows 533-545. This confirms that the speed should be 180 MHz.

```
#if defined (STM32F427_437xx) || defined (STM32F429_439xx)
/* Enable the Over-drive to extend the clock frequency to 180 Mhz */
PWR->CR |= PWR_CR_ODEN;
while((PWR->CSR & PWR_CSR_ODRDY) == 0)
{
}
PWR->CR |= PWR_CR_ODSWEN;
while((PWR->CSR & PWR_CSR_ODSWRDY) == 0)
{
}
/* Configure Flash prefetch, Instruction cache, Data cache and wait state */
FLASH->ACR = FLASH_ACR_PRFTEN | FLASH_ACR_ICEN |FLASH_ACR_DCEN |FLASH_ACR_LATENCY_5WS;
#endif /* STM32F427_437x || STM32F429_439xx */
```

At micro level I am doing fast pulses that should be around 10 nanosecond long, but are 35 nanosecond long as shown in oscilloscope.



This pulse train is generated with the following code

```
#include <stm32f4xx.h>
#include <stm32f4_init.h>

int main(void)
{
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN; // Enable GPIOC clock

    GPIOC->MODER = GPIO_Mode_OUT <<(2*13); // Configure PC13
    GPIOC->OTYPER = GPIO_OType_PP <<(2*13); // ignore other pins
    GPIOC->OSPEEDR = GPIO_Speed_100MHz <<(2*13);
    GPIOC->PUPDR = GPIO_PuPd_NOPULL <<(2*13);

    while (1) {
        GPIOC->BSRRL = GPIO_BSRR_BS_13; // Set PC13
        GPIOC->BSRRH = GPIO_BSRR_BS_13; // Clear PC13
```

```

GPIOC->BSRRL    = GPIO_BSRR_BS_13;
GPIOC->BSRRH    = GPIO_BSRR_BS_13;

}

}

```

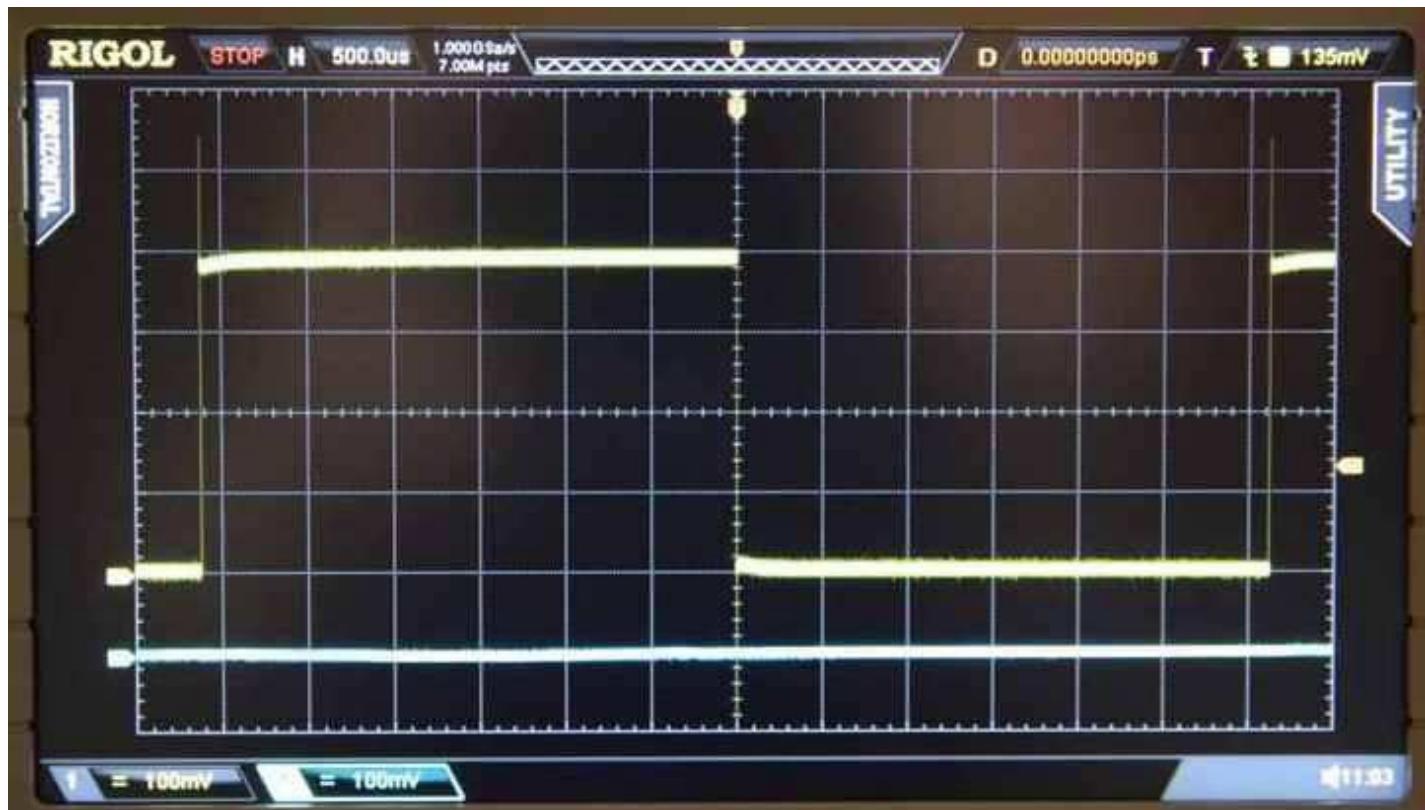
Assembly code for the while(1) loop

```

080002FE  strh    r3, [r2, #24]
08000300  strh    r3, [r2, #26]
08000302  strh    r3, [r2, #24]
08000304  strh    r3, [r2, #26]
08000306  strh    r3, [r2, #24]
08000308  strh    r3, [r2, #26]
0800030A  strh    r3, [r2, #24]
0800030C  strh    r3, [r2, #26]
0800030E  strh    r3, [r2, #24]
08000310  strh    r3, [r2, #26]
08000312  b.n 0x80002fe <main+50>

```

At macro level, I am trying to toggle a pulse with 1000 Hz frequency. The expected pulse width is 1000 microseconds, but oscilloscope shows 3222 microsecond long pulses.



This pulse train was created with the following code.

```

#include <stm32f4xx.h>
#include <stm32f4_init.h>

#define oneMilliSecond 180*1000

int main(void)
{

```

```

uint32_t      triggerLimit=0;
uint32_t      loopCount=0;

RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN; // Enable GPIOC clock

GPIOC->MODER = GPIO_Mode_OUT    <<(2*13); // Configure PC13
GPIOC->OTYPER = GPIO_OType_PP   <<(2*13); // ignore other pins
GPIOC->OSPEEDR = GPIO_Speed_100MHz <<(2*13);
GPIOC->PUPDR = GPIO_PuPd_NOPULL <<(2*13);

CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk; // Enable core debugger for tracing
DWT->CYCCNT = 0; // Reset debugger trace counter
DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk; // Enable cycle counter

while (1) {
if (DWT->CYCCNT > triggerLimit) {
    triggerLimit += oneMilliSecond;
    loopCount++;

    if (loopCount & 1) {
        GPIOC->BSRRL = GPIO_BSRR_BS_13; // Set PC13
    } else {
        GPIOC->BSRRH = GPIO_BSRR_BS_13; // Clear PC13
    }
}
}
}

```

The assembly code for the infinite loop is

08000320	ldr r3, [r1, #4]
08000322	cmp r2, r3
08000324	bcs.n 0x8000320 <main+84>
08000326	adds r0, #1
08000328	lsls r3, r0, #31
0800032A	add.w r2, r2, #179200 ; 0x2bc00
0800032E	add.w r2, r2, #800 ; 0x320
08000332	ite mi
08000334	strhmi r4, [r5, #24]
08000336	strhpl r4, [r5, #26]
08000338	b.n 0x8000320 <main+84>

Note how the 1 millisecond value (180,000) is implemented as 179,200 + 800.

I have observed similar results with the SysTick_Handler configured by SysTick_Config.