Проблема: настроил тактирование CMX CUBE в провел экспорт проекта в VisualGDB – тактовая частота 64 000 000 вместо ожидаемых 168 000 000. Запустил экспорт этого же проекта в IAR – все работает.

Оказалось проблема в конфигурировании при отладке.

Hello,

I have created an application using CubeMX for a STM32f446. I have exported it as MDK-ARM and imported the project in VisualGDB.

The clocks are setup to use HSE, but when I start the application it uses the HSI clock @64MHz. I have also experimented by re-configuring CubeMX to use HSI clock and change the frequencies but no matter what I set everything seems to get ignored and the application always runs at 64MHz.

When I compile the same application in Keil uVision everything is working as expected – all clocks are set up correctly and HSE clock also works fine.

It looks like VisualGDB is not importing something from the CubeMX generated files. Any ideas?

I have modified stm32f4x.cfg from openocd and commented of the following lines. It seems to work now.

Is there any better solution for this?

```
$_TARGETNAME configure -event reset-init {
# Configure PLL to boost clock to HSI x 4 (64 MHz)
#mww 0x40023804 0x08012008 ;# RCC_PLLCFGR 16 Mhz /8 (M) * 128 (N) /4(P)
#mww 0x40023C00 0x00000102 ;# FLASH_ACR = PRFTBE | 2(Latency)
#mmw 0x40023800 0x01000000 0 ;# RCC_CR |= PLLON
#sleep 10 ;# Wait for PLL to lock
#mmw 0x40023808 0x00001000 0 ;# RCC_CFGR |= RCC_CFGR_PPRE1_DIV2
#mmw 0x40023808 0x00000002 0 ;# RCC_CFGR |= RCC_CFGR_SW_PLL

# Boost JTAG frequency
#adapter_khz 8000
adapter_khz 2000
```

Hi,

Thanks for finding this out. Normally, the clock-related code in **stm32fxxx.cfg** would get executed just after starting a debug session. It would raise the system clock frequency, allowing for faster FLASH programming and faster JTAG communication, and once the **SystemClock_Config()** runs, it would override the parameters set via **stm32fxxx.cfg**.
The clock parameters set by the **stm32fxxx.cfg** would only affect the code that runs before **SystemClock_Config()** (i.e. startup code) and should not affect the rest of the program.
Either way, if you prefer disabling this mechanism, you can copy a modified version of the .cfg file to the project directory and replace the path to it in OpenOCD settings (expand the Advanced view to edit it) to **$(ProjectDir.forwardslashes)/<cfg file name>**.
Regarding the startup file, VisualGDB normally does not allow replacing it because the file shipped with VisualGDB is 100% equivalent to the original .s file. It only defines the interrupt vector table and the reset handler and does not handle the clock. If you absolutely need to override it, please consider setting the “excluded from build” flag for the .c file and adding the .s file to the project manually.

Hello,

First of all, thank you Artem for your posts, it helped me to find out what is the issue in my case

I had similar issue with STM32L471. It turned out, that problem was not with changing of MSI clock (from 4MHz to 24MHz to boost jtag) by debugger, but with disabling instruction cache when writing into FLASH_ACR register (in order to set flash latency and enable prefetch). My init code didn't set cache on/off at all, and therefore I had different behavior when starting using debugger (which disabled instr. cache) compared to restart by application itself (which kept instr. cache enabled as this is the default state after reset).

So maybe not that much related to your issue Artem, but maybe it's worth to check if you are configuring everything, what was touched by debuger, in your init code (even something not that much related to clock as instruction cache).

Here is the part of openocd cfg (stm32l4x.cfg in my case), which was modifying cache configuration – first mww instruction touching FLASH_ACR register:

```
1 $_TARGETNAME configure -event reset-init {
2         # CPU comes out of reset with MSI_ON | MSI_RDY | MSI Range 6 (4 MHz).
3         # Use MSI 24 MHz clock, compliant even with VOS == 2.
4         # 3 WS compliant with VOS == 2 and 24 MHz.
5         mww 0x40022000 0x00000103   ;# FLASH_ACR = PRFTBE | 3(Latency)
6         mww 0x40021000 0x00000099   ;# RCC_CR = MSI_ON | MSIRGSEL | MSI Range 9
7         # Boost JTAG frequency
8         adapter_khz 4000
9 }
```

Regards

Jan

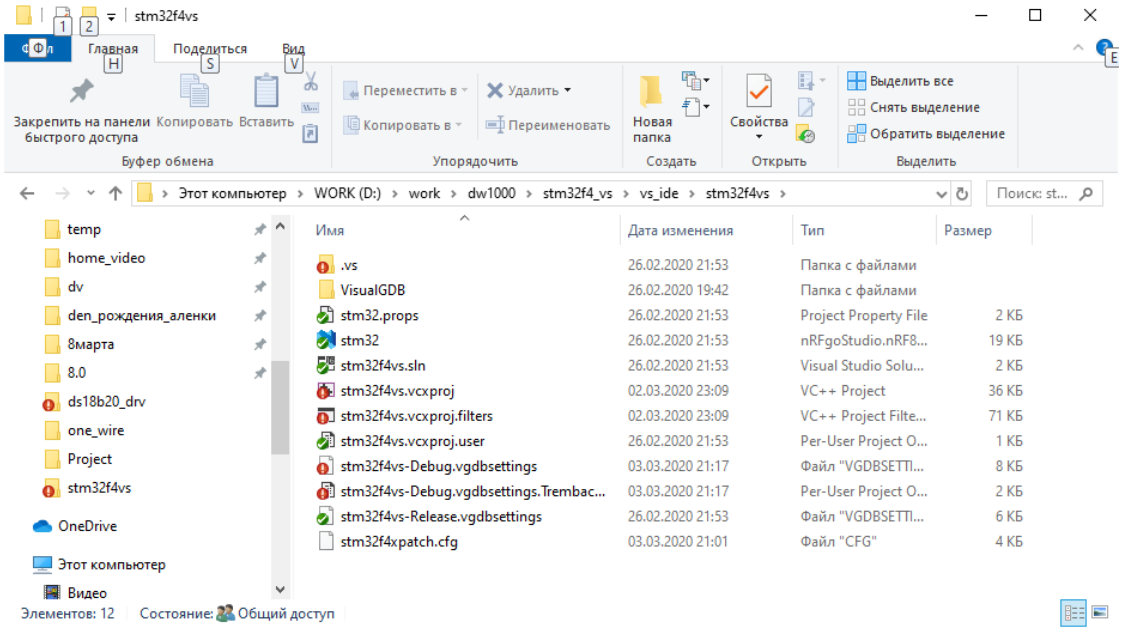
Было:

```
$_TARGETNAME configure -event reset-init {
        # Configure PLL to boost clock to HSI x 4 (64 MHz)
        mww 0x40023804 0x08012008   ;# RCC_PLLCFGR 16 Mhz /8 (M) * 128 (N) /4(P)
        mww 0x40023C00 0x00000102   ;# FLASH_ACR = PRFTBE | 2(Latency)
        mmw 0x40023800 0x01000000 0 ;# RCC_CR |= PLLON
        sleep 10                    ;# Wait for PLL to lock
        mmw 0x40023808 0x00001000 0 ;# RCC_CFGR |= RCC_CFGR_PPRE1_DIV2
        mmw 0x40023808 0x00000002 0 ;# RCC_CFGR |= RCC_CFGR_SW_PLL

        # Boost JTAG frequency
        adapter_khz 8000
}
```

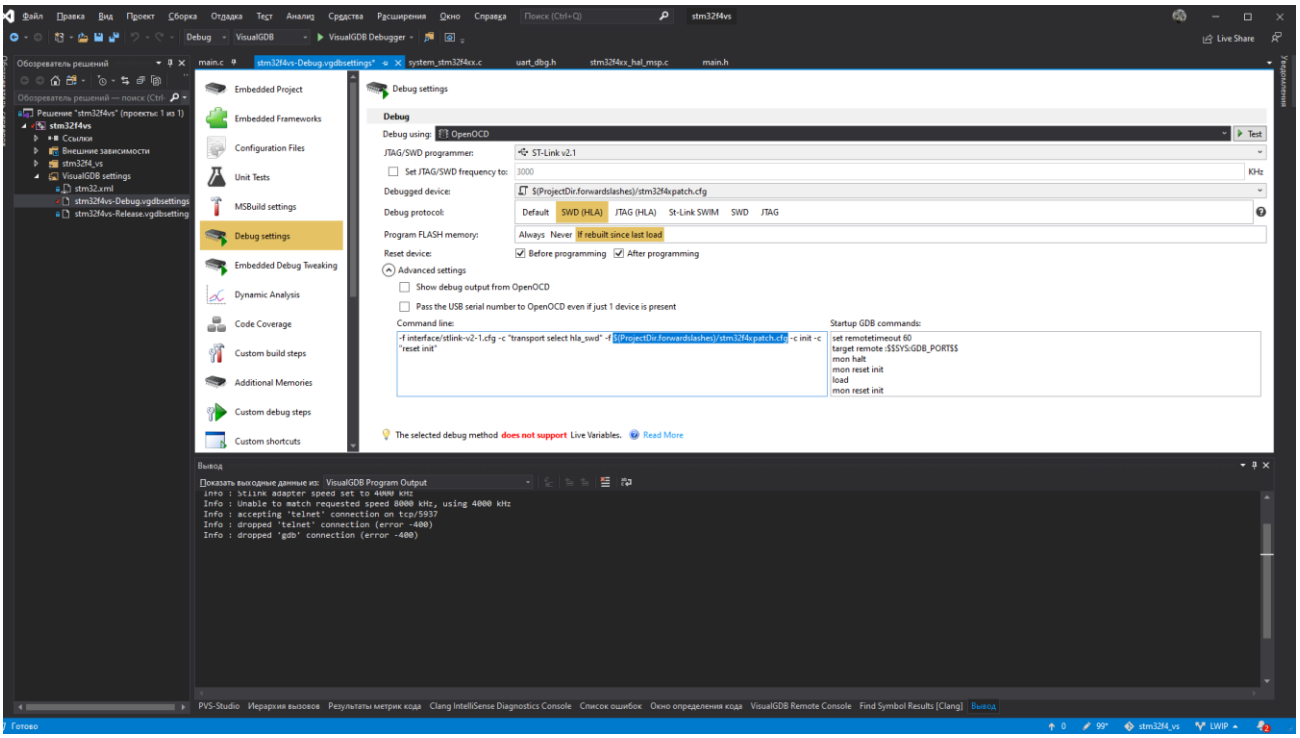
Стало:

```
$_TARGETNAME configure -event reset-init {
        # Configure PLL to boost clock to HSE   (168 MHz)
        mww 0x40023804 0x04405408   ;# RCC_PLLCFGR 8 Mhz /8 (M) * 336 (N) /2(P)
        mww 0x40023C00 0x00000105   ;# FLASH_ACR = PRFTBE | 5(Latency)
        mmw 0x40023800 0x01000000 0 ;# RCC_CR |= PLLON
        sleep 10                    ;# Wait for PLL to lock
        mmw 0x40023808 0x00001000 0 ;# RCC_CFGR |= RCC_CFGR_PPRE1_DIV2
        mmw 0x40023808 0x00000002 0 ;# RCC_CFGR |= RCC_CFGR_SW_PLL

        # Boost JTAG frequency
        adapter_khz 8000
}
```

Положил файл в папку проекта:



Поменял настройки:



-f interface/stlink-v2-1.cfg -c "transport select hla_swd" -f $(ProjectDir.forwardslashes)/stm32f4xpatch.cfg -c init -c "reset init"