

# Микроконтроллеры STM32

Домашнее задание №4

Евгений Зеленин

10 марта 2025 г.

## 1. Постановка задачи

**Условия задачи.** Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Инициализируйте перезагрузку микроконтроллера по нажатию на клавишу Reset и по срабатыванию сторожевого таймера IWDG. Считайте состояние регистра RCC\_CSR и определите причину перезагрузки микроконтроллера. Целью домашней работы является получение учащимися опыта по определению источника перезагрузки микроконтроллера, использованию и конфигурированию сторожевых таймеров. В качестве результата выполнения ДЗ представьте файл main.c созданного проекта. В комментарии укажите, какие биты были выставлены в регистре RCC\_CSR при различных типах перезагрузки микроконтроллера.

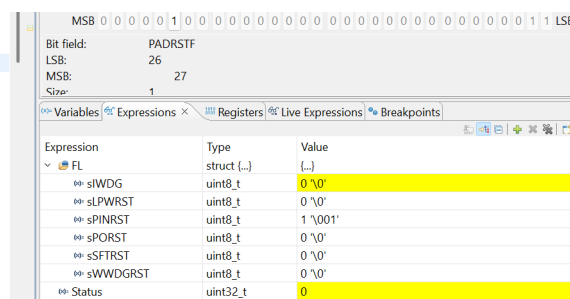
## 2. Статусы регистра CSR при перезагрузке

Код, используемые в этой работе, приведен в файле main.c в материалах к занятию. В этом разделе будут показаны флаги, полученные во время перезагрузки МК разными способами (Рисунки 1, 2, 3).

```
-----
FL.SFSTRST = LL_RCC_IsActiveFlag_SFTRST();
FL.SWWDGRST = LL_RCC_IsActiveFlag_WWDGRST();
LL_RCC_ClearResetFlags();
uint32_t Status = RCC->CSR;
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
HAL_IWDG_Refresh(&hiwdg);
HAL_Delay(300);
while (1) {
    HAL_IWDG_Refresh(&hiwdg);
    HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);

/* USER CODE END WHILE */

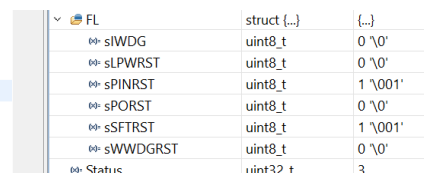
/* USER CODE BEGIN 3 */
    HAL_Delay(300);
    HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
}
```



Expression	Type	Value
FL	struct [...]	[...]
sIWDG	uint8_t	0 \0'
sLPWRST	uint8_t	0 \0'
sPINRST	uint8_t	1 \001'
sPORST	uint8_t	0 \0'
sSFTRST	uint8_t	0 \0'
sWWDGRST	uint8_t	0 \0'
Status	uint32_t	0

Рис. 1: Перезагрузка кнопкой reset

```
/* USER CODE BEGIN 3 */
    HAL_Delay(400);
    HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
    HAL_NVIC_SystemReset();
}
/* USER CODE END 3 */
```



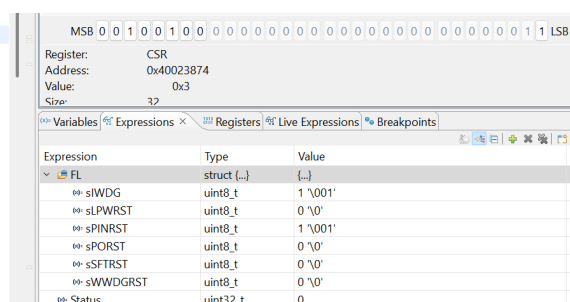
Expression	Type	Value
FL	struct [...]	[...]
sIWDG	uint8_t	0 \0'
sLPWRST	uint8_t	0 \0'
sPINRST	uint8_t	1 \001'
sPORST	uint8_t	0 \0'
sSFTRST	uint8_t	1 \001'
sWWDGRST	uint8_t	0 \0'
Status	uint32_t	3

Рис. 2: Программный сброс

```
FL.SWWDGRST = LL_RCC_IsActiveFlag_WWDGRST();
LL_RCC_ClearResetFlags();
uint32_t Status = RCC->CSR;
HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_RESET);
HAL_IWDG_Refresh(&hiwdg);
HAL_Delay(507);
while (1) {
    HAL_IWDG_Refresh(&hiwdg);
    HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);

/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    HAL_Delay(500);
    HAL_GPIO_WritePin(LED2_GPIO_Port, LED2_Pin, GPIO_PIN_SET);
    HAL_NVIC_SystemReset();
}
```



Expression	Type	Value
FL	struct [...]	[...]
sIWDG	uint8_t	1 \001'
sLPWRST	uint8_t	0 \0'
sPINRST	uint8_t	1 \001'
sPORST	uint8_t	0 \0'
sSFTRST	uint8_t	0 \0'
sWWDGRST	uint8_t	0 \0'
Status	uint32_t	0

Рис. 3: Перезагрузка с помощью watchdog

Как видно из рисунка 1, в случае сброса с помощью кнопки (внешний reset), в регистре CSR установлен только флаг sPINRST. В случае программного сброса (рисунок 2), установлены флаги sPINRST и sSFRST. При сбросе с помощью watchdog, в регистре CSR будут установлены флаги sPINRST и sIWDG.

Исходя из полученных флагов, можно сделать вывод, что флаг sPINRST появляется в любом случае после сброса МК.

### 3. Интересные наблюдения

Также, во время работы было потрачено некоторое время на эксперименты с watchdog и отладкой. Оказалось, что установка breakpoint вне области выполнения (допустим, за бесконечным циклом), приводит к некоторым задержкам в выполнении программы. В то же время, watchdog работает независимо и продолжает считать. Может возникнуть ситуация, когда счетчик watchdog не успеет сброситься из-за этой задержки и МК перезагрузится. Подробно этот эффект показан в видео к занятию.

Проверим на практике. Подключим к плате два светодиода: первый загорается сразу после инициализации до бесконечного цикла, гаснет только в конце основного цикла и больше не загорается. Второй светодиод все время моргает, но только когда выполнение дойдет до основного цикла. Задержка в цикле такая же - 450мс. Сброс watchdog производится в начале цикла.

Далее, если установить breakpoint до бесконечного цикла while(), когда выполнение уже дошло до работы основного цикла (т.е. когда первый диод потух, а второй моргает), то в этот момент контроллер перезагружается по watchdog, и устанавливается флаг sLWDG. При сбросе через кнопку reset все аналогично - будет флаг sLWDG. Визуально определить это просто - после запуска МК загорается первый светодиод, потом начинает моргать второй и при установке breakpoint эта последовательность повторяется.

Затем, если изменить задержку в основном цикле, допустим, на 350мс, то все работает как и должно, во время "мигания" установка breakpoint до бесконечного цикла while() не сбрасывает МК. Теперь по сбросу через reset устанавливается только флаг sPINRST.

Чтобы избежать такого неопределенного поведения, есть возможность остановить watchdog во время отладки. Для этого можно воспользоваться двумя способами:

- Включить настройку "Suspend watchdog while halted" (Рисунок 4)
- Использовать команду \_\_HAL\_DBGMCU\_FREEZE\_IWDG()

При использовании любого из этих методов, описанное выше неопределенное поведение исчезает.

Также отмечу, что для МК STM32F401CCU6 задержка на установку первого брейкпоинта составляет 76мс, а второго и последующих - 60мс.

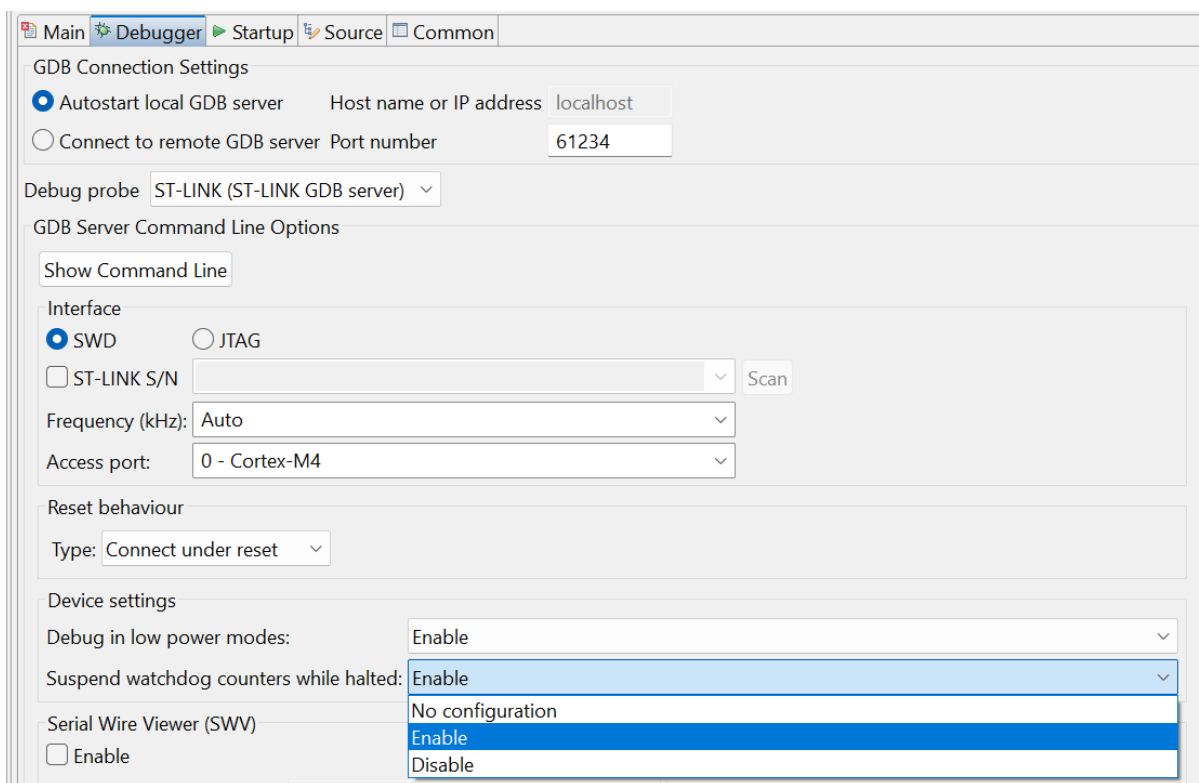


Рис. 4: Остановка watchdog во время отладки

#### 4. Дополнительные материалы

Демонстрация работы и материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №04