Микроконтроллеры STM32

Домашнее задание 6

Евгений Зеленин

19 марта 2025 г.

1. Постановка задачи

Условия задачи. *Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Продемонстрируйте переход микроконтроллера в режим сна и последующий выход (продолжение работы) в качестве события используя нажатие на кнопку, либо событие таймера. Целью домашнего задания является закрепление знаний полученных в ходе вебинара и получение учащимися практического опыта перевода микроконтроллеров в режим с пониженным энергопотреблением. В качестве результата работы представьте файл main.c созданного вами проекта. Кратко опишите достигнутые результаты.*

2. Режим PVD

Сначала, продемонстрируем режим работы PVD. Для этого нужно настроить прерывание PVD_Interrupt, сконфигурировать PVD и затем запустить. Прерывание можно настроить как из интерфейса CubeMX (рисунок 1).

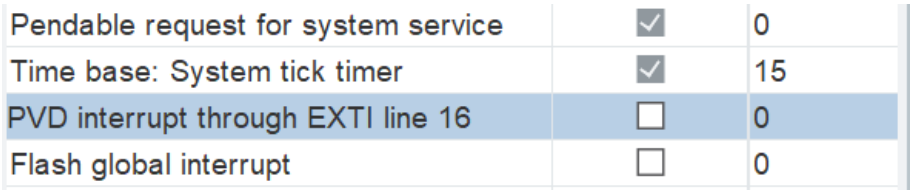

Рис. 1: Настройка прерывания PVD

Так и непосредственно при инициализации МК, добавив следующие строчки:.

```
 HAL_NVIC_SetPriority(PVD_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(PVD_IRQn);
```

Конфигурирование и запуск осуществляется следующим образом:

```
PWR_PVDTypeDef sConfigPVD;

sConfigPVD.PVDLevel = PWR_CR_PLS_LEV7;
sConfigPVD.Mode = PWR_PVD_MODE_IT_RISING_FALLING;

HAL_PWR_ConfigPVD(&sConfigPVD);
HAL_PWR_EnablePVD();
```

Далее, необходимо добавить функцию обратного вызова PVD.

```
void HAL_PWR_PVDCallback(void) {
 HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);

}
```

Если PVD_Interrupt было включено без использования CubeMX, то потребуется добавить еще следующие строки:

```
  void PVD_IRQHandler(void)
{
  HAL_PWR_PVD_IRQHandler();
}
```

Алгоритм работы заключается в следующем: при пересечении питающим напряжением пороговых уровней, генерируется прерывание и вывод микроконтроллера изменяет свое состояние на противоположное. В нашем случае, уровень и границы задаются с помощью sConfigPVD . PVDLevel = PWR_CR_PLS_LEV7, а режим переключения с помощью sConfigPVD . Mode = PWR_PVD_MODE_IT_RISING_FALLING.

Видео с демонстрацией работы приложено в материалах к занятию.

3. Режим SleepOnExit

Вход в режим Sleep On Exit осуществляется схожей по названию функцией HAL_PWR _EnableSleepOnExit(). После выполнения которой, микроконтроллер переходит в режим сна до тех пор, пока не возникнет прерывание. После выхода из прерывания, контроллер опять перейдет в режим сна.

Чтобы продемонстрировать этот режим работы сделаем следующее: запустим SleepOnExit непосредственно перед бесконечным циклом, а в сам цикл поместим код для мигания светодиодом с частотой 1Гц.

Подключим к МК кнопку и в теле обработчика прерывания но нажатию будем мигать светодиодом 5 раз с частотой 2.5 Гц.

Таким образом, если все настроено верно, то код мигающий диодом с частотой 1 Гц никогда не должен выполняться. После включения МК сразу уйдет в сон, по нажатию на кнопку проснется, моргнет 5 раз светодиодом и опять уйдет в сон.

Стоит отметить, что для использования функции HAL_Delay() в прерывании потребовалось изменить приоритеты прерываний, например следующим образом:

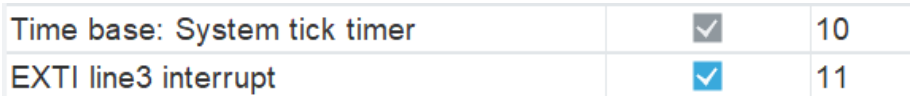

Рис. 2: Настройка приоритета прерываний

В противном случае, прерывание системного тика не сможет вытеснить обработчик прерывания по нажатию кнопки.

Также, в место HAL_Delay можно просто использовать цикл и зная частоту МК рассчитать необходимое количество циклов для нужной задержки. Видео с демонстрацией работы приложено в материалах к занятию.

4. Дополнительные материалы

Материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №06