Микроконтроллеры STM32

Домашнее задание 7

Евгений Зеленин

23 марта 2025 г.

1. Постановка задачи

Условия задачи.

Задание 1. Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Подключите к отладочной плате светодиод и продемонстрируйте циклическое изменение яркости этого светодиода посредством генерации PWM сигнала переменной скважности. Значения скважности сигнала обновляйте используя DMA из заранее сформированного массива значений. Установите частоту мигания светодиода не более 1-2 Гц.

Задание 2. Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Подключите к отладочной плате делитель напряжения одним из элементов которого является фоторезистор. При помощи DMA организуйте автоматическую отправку данных об освещенности в массив. Целью домашнего задания является получение учащимся практических навыков работы с DMA. В качестве результата требуется прислать 2 коротких видео демонстрирующих работу созданного кода.

2. Мигание светодиодом

Сделаем два варианта мерцания светодиода. Первый вариант - изменение коэффициента заполнения шим с помощью DMA при частоте шим в 1Гц .

Кода совсем немного:

```
uint16_t pwmval[10] = {0,99,199,299,399,499,599,699,799,899};
HAL_TIM_PWM_Start_DMA(&htim1, TIM_CHANNEL_1, (const uint32_t *) pwmval, 10);
```

Второй вариант - частота ШИМ 100 Гц, для плавного изменения коэффициента заполнения воспользуемся циклом и сформируем массив из 100 значений, затем запустим DMA и укажем в качестве параметра на запуск DMA адрес первого элемента массива:

```
for(uint8_t i = 0; i < 50; i++){
 pwmval[i] = i*2;
}
for(uint8_t i = 50; i < 100; i++){
  pwmval[i] = (99 - i) * 2;
 }
HAL_TIM_PWM_Start_DMA(&htim1, TIM_CHANNEL_1, (const uint32_t *) pwmval, 100);
```

В материалах к занятию приложены видео в режиме SlowMo и с обычной скоростью.

3. DMA и ADC

Также сделаем два варианта этого ДЗ.

Первый вариант - АЦП запускается по событию таймера 2, производит одно преобразование и с помощью DMA кладет результат измерения в массив. Когда весь массив заполнен, вызывается прерывание и процесс запускается заново. Для индикации работы, сделано прерывание таймера по событию переполнения, в котором изменяется состояние светодиода на противоположное. Частота таймера 10Гц. Когда тоже совсем немного.

```
  HAL_ADC_Start_DMA(&hadc1, (uint32_t *)  adcvals, 50);
  HAL_TIM_Base_Start_IT(&htim2);

  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
   HAL_ADC_Start_DMA(&hadc1, (uint32_t *) adcvals, 50);

}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
 HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
}
```

Теперь, усложним задачу и сделаем так, чтобы светодиод мигал без использования прерываний и процессорного времени.

Для этого сделаем следующие настройки:

DMA Request	Stream	Direction	Priority
ADC1	DMA2 Stream 0	Peripheral To Memory	Low
TIM1_UP	DMA2 Stream 5	Memory To Peripheral	Low

Рис. 1: Настройка каналов DMA

Преобразование АЦП будет запускаться по событию Capture Compare 2 таймера Timer1 и с помощью DMA помещать результат единичного преобразования в массив. В то же время, по событию переполнения счетчика таймера 1 будет создаваться запрос на копирование данных из памяти в периферию. DMA можно использовать для заполнения регистра BSRR порта GPIOA значениями из массива blink[2]. Если в этот массив записать требуемые состояния регистра, можно добиться мигания светодиода с частотой равной частоте таймера.

Для запуска DMA с нужными параметрами используем функцию HAL_DMA_Start().

Чтобы включить DMA по событию переполнения воспользуемся макросом __HAL_TIM _ENABLE_DMA().

Вся суть сводится к нескольким строчкам кода:

```
  uint32_t blink[2] = {LED_Pin, LED_Pin << 16};

  HAL_ADC_Start_DMA(&hadc1, (uint32_t *) adcval, 50);
```

```
  uint32_t addr = (uint32_t)&LED_GPIO_Port->BSRR;
  HAL_DMA_Start(&hdma_tim1_up, (uint32_t) blink, (uint32_t) addr, 2);
  __HAL_TIM_ENABLE_DMA(&htim1, TIM_DMA_UPDATE);
  HAL_TIM_Base_Start(&htim1);
  HAL_TIM_OC_Start(&htim1, TIM_CHANNEL_2);
```

После запуска этого кода процессор будет прерываться только когда массив полностью заполнится данными.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
{
 HAL_ADC_Start_DMA(&hadc1, (uint32_t *) adcval, 50);
}
```

На рисунке ниже показан массив значений, полученных с АЦП с помощью DMA. Время заполнения всего массива целиком - около 5 секунд, такое время выбрано чтобы можно было закрыть фоторезистор рукой и наглядно показать изменение результатов АЦП преобразования во времени. Демонстрация работы приложена к материалам занятия.

adcvals[12]	uint16_t	88
adcvals[13]	uint16_t	150
adcvals[14]	uint16_t	367
adcvals[15]	uint16_t	464
adcvals[16]	uint16_t	474
adcvals[17]	uint16_t	525
adcvals[18]	uint16_t	224
adcvals[19]	uint16_t	70
adcvals[20]	uint16_t	68
adcvals[21]	uint16_t	66
adcvals[22]	uint16_t	68
adcvals[23]	uint16_t	70
adcvals[24]	uint16_t	66
adcvals[25]	uint16_t	68
adcvals[26]	uint16_t	68
adcvals[27]	uint16_t	67
adcvals[28]	uint16_t	71
adcvals[29]	uint16_t	246
adcvals[30]	uint16_t	557
adcvals[31]	uint16_t	705
adcvals[32]	uint16_t	800
adcvals[33]	uint16_t	910
adcvals[34]	uint16_t	1097
adcvals[35]	uint16_t	1448
adcvals[36]	uint16_t	1708
adcvals[37]	uint16_t	1804
adcvals[38]	uint16_t	1796
adcvals[39]	uint16_t	272
adcvals[40]	uint16_t	71
adcvals[41]	uint16_t	68
adcvals[42]	uint16_t	69
adcvals[43]	uint16_t	66
adcvals[44]	uint16_t	66

Рис. 2: Массив, заполненный значениями

4. Дополнительные материалы

Материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №07