Генерация сигналов с помощью таймера STM32L4, DMA и DAC

Опубликовано 2019-03-30

Генерация произвольных сигналов с помощью микроконтроллера чрезвычайно полезна. Его можно использовать, например, для воспроизведения любого звука или создания модулятора для модема. Наиболее необходимым периферийным устройством MCU является, конечно, ЦАП, но ему также нужны другие периферийные устройства для эффективного воспроизведения сэмплов без загрузки процессора.

Этот пост показывает, как реализовать генератор сигналов на STM32L432 без использования библиотек HAL.

Вы можете найти аналогичный пример для AVR XMEGA здесь.

Первое, конечно, чтобы иметь аудио образцы, хранящиеся где-то. В этом примере используется 77 образцов одного цикла sinewave (продолжайте читать, почему именно 77). STM32L432 DAC можно использовать в 8-битном режиме, поэтому каждый пример является просто uint8_t и хранится во flash.

Цель состоит в том, чтобы генерировать тоны 2200 и 1200 Гц. Таким образом, образцы должны обновляться 1200*77 и 2200*77 раз в секунду (потому что 77 образцов составляют один цикл). Для более высокого тона это фактически означает частоту обновления 169400 Гц, что приведет к слишком большой нагрузке прерываний на процессор, поэтому лучше всего обрабатывать его с помощью DMA. DMA просто копирует последовательные образцы из flash в выходной адрес регистра ЦАП. DMA однако не знает **когда** чтобы скопировать образцы (это может идти так быстро, как позволяют подсистемы шины и памяти), поэтому еще одно периферийное устройство должно сказать DMA, когда делать передачу. В данном примере это таймер общего назначения TIM2.

Не все таймеры могут генерировать запросы DMA. Это проиллюстрировано в справочном руководстве. В моем случае я просто взял канал DMA1 1 (есть две отдельные периферии DMA, каждая со своими собственными каналами), что означает, что я должен использовать timer TIM2 channel 3 (не путайте DMA channel с timer output compare channel).

Еще одна проблема, о которой необходимо позаботиться, - это архитектура памяти. В случае STM32L4 это не проблема, но в случае других STM32 вы можете быть более ограничены (например, только DMA2 может получить доступ к области памяти ЦАП). Обычно это показано в одной из первых глав руководства:

Цель состоит в том, чтобы переместить данные из flash в DAC. ЦАП подключен к APB1, который подключен к AHB1 (AHB-это “быстрая” системная шина, APB-более медленная шина). Как вы можете видеть, и DMA1, и DMA2 могут получить доступ к флэш-памяти, и оба они также могут получить доступ к пространству памяти AHB1.

Подведение итогов:

- Таймер TIM2 выходной канал 3 будет тикать с соответствующей частотой обновления
- Выходной канал TIM2 3 сгенерирует событие переполнения
- Это событие вызовет передачу DMA1 по каналу 1
- DMA1 channel 1 будет передавать данные со вспышки на 8-битный выходной регистр ЦАП
- Канал DMA1 1 будет использовать круговой режим, поэтому, когда он достигнет последнего образца, он начнет с самого начала
- ЦАП ничего не знает, он только должен быть включен и будет обновлять напряжение на каждой записи выходного регистра

Код

Вот код. Он не использует библиотеки HAL и не имеет дополнительных зависимостей. Частота APB1 составляет 10 МГц (это актуально для TIM2).

```
#include <stm32l432xx.h>

static const uint8_t SINEWAVE[] =
{ 126, 136, 146, 156, 166, 175, 185, 194, 202, 210, 217, 224, 230, 235, 240, 244, 247, 249, 251, 251, 251, 250, 248, 246, 242,
    238, 233, 227, 221, 214, 206, 198, 189, 180, 171, 161, 151, 141, 131, 121, 111, 101, 91, 81, 72, 63, 54, 46, 38, 31, 25, 19,
    14, 10, 6, 4, 2, 1, 1, 1, 3, 5, 8, 12, 17, 22, 28, 35, 42, 50, 58, 67, 77, 86, 96, 106, 116 };

#define TIMER_RELOAD_FOR_2200Hz (2 * 59)
#define TIMER_RELOAD_FOR_1200Hz (2 * 108)

void dac_play_sinewave(void)
{

  //pin initialization is done in pins.h
  RCC->APB1ENR1 |= RCC_APB1ENR1_DAC1EN_Msk; //enable clock to the DAC

  DAC->CR = DAC_CR_EN1; //enable DAC channel 1

  RCC->AHB1ENR |= RCC_AHB1ENR_DMA1EN_Msk; //enable clock to DMA1 peripheral

  //set up DMA transfer, priority low, source size 8-bit, destination size 8-bit,
  DMA1_Channel1->CCR =
  DMA_CCR_MINC_Msk/*increment source memory address*/|
  DMA_CCR_CIRC_Msk/*circular mode*/|
```

```
  DMA_CCR_DIR_Msk/*direction from memory to peripheral*/;

  DMA1_Channel1->CNDTR = sizeof(SINEWAVE) / sizeof(SINEWAVE[0]); //how many data to transfer
  DMA1_Channel1->CMAR = (uint32_t) &SINEWAVE[0]; //source address - sine wave table
  DMA1_Channel1->CPAR = (uint32_t) &(DAC->DHR8R1); //destination address - DAC output register

  DMA1_CSELR->CSELR |= (0b0100 << DMA_CSELR_C1S_Pos); //TIM2_CH3 triggers DMA1 channel 1

  DMA1_Channel1->CCR |= DMA_CCR_EN_Msk; //enable DMA channel

  //TIM2 is clocked from APB1 at 10MHz
  RCC->APB1ENR1 |= RCC_APB1ENR1_TIM2EN_Msk; //enable clock to TIM2

  TIM2->CR1 = 0; //stop the timer
  TIM2->CNT = 0; //reset timer value
  TIM2->ARR = TIMER_RELOAD_FOR_2200Hz;

  TIM2->DIER = TIM_DIER_CC3DE_Msk; //enable timer channel 3 DMA request (goes to DMA channel 1)
  TIM2->EGR = TIM_EGR_CC3G_Msk; //channel 3 can generate events

  TIM2->CR1 = TIM_CR1_CEN_Msk; //start the timer
}

void dac_shift_frequency(void)
{
  //this will alternate the ARR register between words for 1200 and 2200 Hz
  TIM2->ARR = (TIMER_RELOAD_FOR_2200Hz + TIMER_RELOAD_FOR_1200Hz) - TIM2->ARR;
}

void dac_deinit(void)
{

  TIM2->CR1 = 0; //stop the timer
  DMA1_Channel1->CCR = 0; //disable DMA channel
  DAC->DHR8R1 = 0; //ouput 0 code
  DAC->CR = 0; //disable DAC

  RCC->APB1ENR1 &= ~(RCC_APB1ENR1_DAC1EN_Msk | RCC_APB1ENR1_TIM2EN_Msk); //disable clock to the DAC and TIM2
}
```

Магические числа

Есть некоторые магические числа, такие как 77 образцов синусоидальной волны, значения перезагрузки таймера 59 и 108. Откуда они взялись? Прежде всего – значение перезагрузки таймера. Он определяет, как долго таймер должен считать, пока он не переполнится и не начнет снова с нуля (поэтому чем выше значение перезагрузки, тем ниже частота дискретизации и ниже выходная частота). Таймер APB1 работает с тактовой частотой 10 МГц. Для генерации 1200 Гц с использованием 77 образцов они должны обновляться со скоростью 1200 * 77 Гц, т. е. 92400 Гц. Таймер должен переполняться с такой скоростью, поэтому, чтобы вычислить слово перезагрузки, разделите 10’000’000 Гц на 92400 Гц, и вы получите 108,225… Конечно, регистр перезагрузки таймера принимает только целочисленные значения, поэтому он должен быть равен 108. Разница между идеальным значением и тем, которое может быть введено в регистр, приведет к небольшой частотной ошибке.

Чтобы минимизировать ошибку, я составил простую электронную таблицу, которая рассчитала среднюю ошибку для обоих тонов в зависимости от количества выборок, и выяснил, что для часов APB с частотой 10 МГц минимальная ошибка(0,2%) составляет 77 выборок.

После записи звука с помощью звуковой карты и измерения частоты в Audacity я обнаружил, что частоты ровно в два раза выше, поэтому я умножил значения перезагрузки таймера на 2 без дальнейших исследований.