**Лабораторная работа № 7: Работа с DMA**

**Цель работы:** исследовать возможности использования DMA, основные характеристики DMA, взаимодействие с другими устройствами в составе микроконтроллера.

**Оборудование и программное обеспечение**: отладочная плата STM32F4 Discovery, мультиметр, реда разработки CooCox CoIDE 1.7, Keil 4.xx, инструменты построения проекта GNU Toolchain for ARM Embedded Processors, библиотека CMSIS.

**Теоретические сведения**

Direct Memory Access (DMA, рос. “Прямой доступ к памяти”) – механизм, используемый в контроллерах ARM для перемещения данных между памятью и периферией без участия процессора. Ключевым моментом является, то что при использовании DMA на перемещение данных не используются ресурсы процессора, что может быть особенно критичным при создании приложений, работающих с большим количеством данных и активно использующих периферию. Работа DMA обеспечивается отдельным контроллером, который выполняет определенные действия по команде процессора.

Рассматриваемый контроллер имеет в своем распоряжении сразу два контроллера DMA, которые обеспечивают в общей сложности 16 потоков (по 8 потоков на каждый контроллер), каждый из которых предназначен для управления запросами к памяти от одного или нескольких устройств. Каждый поток может обеспечивать до 8 каналов (запросов). Каждый контроллер DMA имеет устройство разрешения конфликтов для обработки запросов в соответствии с их приоритетом.

Контроллер DMA позволяет производить запись данных в трех направлениях:

- от периферии в память;

- из памяти к периферии;

- из памяти в память.

Передача ведется либо в режиме непосредственной передачи, либо в режиме очереди. Поддерживается различный размер данных для передачи, причем размер данных для приемника и источника может быть неодинаковым. В таком случае DMA определяет данную ситуацию и выполняет необходимые действия для оптимизации передачи, однако эта возможность поддерживается только в режиме очереди.

Кроме того, очень полезной может оказаться функция циклической передачи, при использовании которой передача данных начинается с начального адреса снова после передачи последней единицы данных источника.

**Пример программы**

Рассмотрим пример программы, которая использует DMA для передачи данных из памяти в ЦАП. Изменение уровня сигнала на выходе ЦАП происходит циклически с частотой изменения в 1 с. Изменения происходят на основе значения, которое считывается из памяти по сигналу переполнения таймера. В результате этого ЦАП отправляет запрос к DMA и получает данные, которые записываются в регистр данных, что приводит к изменению уровня сигнала. Схема взаимодействия выглядит следующим образом (рис. 3.10):

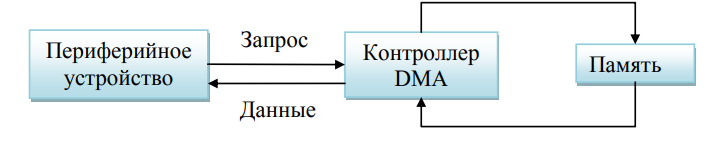


Рис. 3.10. Схема использования DMA

В самой программе следует обратить внимание на большое количество параметров, необходимых для настройки DMA по сравнению с другими устройствами. С этим связана и сложность использования DMA, поскольку необходимо учитывать большое количество возможных настроек. Тем не менее, многие из них достаточно просты (направление передачи, значения адресов), поэтому изучение DMA можно сопоставить по сложности с другими устройствами, в чем очень помогает библиотека Standard Peripheral Library

#include <stm32f4xx.h>

#include <stm32f4xx\_rcc.h>

#include <stm32f4xx\_dma.h>

#include <stm32f4xx\_gpio.h>

#include <stm32f4xx\_tim.h>

#include <stm32f4xx\_dac.h>

uint8\_t levels[] = {0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77};

void init(void);

void init\_gpio(void);

void init\_timer(void);

void init\_dac(void);

void init\_dma(void);

int main(void)

{

init();

while(1)

{

}

}

void init(void) {

init\_gpio();

init\_timer();

init\_dac();

init\_dma();

}

void init\_gpio(void) {

GPIO\_InitTypeDef gpio\_init;

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);

gpio\_init.GPIO\_Mode = GPIO\_Mode\_AN;

gpio\_init.GPIO\_Pin = GPIO\_Pin\_4;

gpio\_init.GPIO\_OType = GPIO\_OType\_PP;

gpio\_init.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

gpio\_init.GPIO\_Speed = GPIO\_Speed\_100MHz;

GPIO\_Init(GPIOA, &gpio\_init);

}

void init\_timer(void) {

TIM\_TimeBaseInitTypeDef tim\_init;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM6, ENABLE);

tim\_init.TIM\_CounterMode = TIM\_CounterMode\_Up;

tim\_init.TIM\_Period = 16000 - 1;

tim\_init.TIM\_Prescaler = 1000 - 1;

TIM\_TimeBaseInit(TIM6, &tim\_init);

TIM\_SelectOutputTrigger(TIM6, TIM\_TRGOSource\_Update);

TIM\_Cmd(TIM6, ENABLE);

}

void init\_dac(void) {

DAC\_InitTypeDef dac\_init;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_DAC, ENABLE);

DAC\_StructInit(&dac\_init);

dac\_init.DAC\_Trigger = DAC\_Trigger\_T6\_TRGO;

dac\_init.DAC\_OutputBuffer = DAC\_OutputBuffer\_Enable;

dac\_init.DAC\_WaveGeneration = DAC\_WaveGeneration\_None;

DAC\_Init(DAC\_Channel\_1, &dac\_init);

}

void init\_dma(void) {

DMA\_InitTypeDef dma\_init;

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_DMA1, ENABLE);

DMA\_DeInit(DMA1\_Stream5);

dma\_init.DMA\_Channel = DMA\_Channel\_7;

dma\_init.DMA\_PeripheralBaseAddr = (uint32\_t)(DAC\_BASE + 0x10);

dma\_init.DMA\_Memory0BaseAddr = (uint32\_t)&levels;

dma\_init.DMA\_DIR = DMA\_DIR\_MemoryToPeripheral;

dma\_init.DMA\_BufferSize = 8;

dma\_init.DMA\_PeripheralInc = DMA\_PeripheralInc\_Disable;

dma\_init.DMA\_MemoryInc = DMA\_MemoryInc\_Enable;

dma\_init.DMA\_PeripheralDataSize = DMA\_PeripheralDataSize\_Byte;

dma\_init.DMA\_MemoryDataSize = DMA\_PeripheralDataSize\_Byte;

dma\_init.DMA\_Mode = DMA\_Mode\_Circular;

dma\_init.DMA\_Priority = DMA\_Priority\_High;

dma\_init.DMA\_FIFOMode = DMA\_FIFOMode\_Disable;

dma\_init.DMA\_FIFOThreshold = DMA\_FIFOThreshold\_HalfFull;

dma\_init.DMA\_MemoryBurst = DMA\_MemoryBurst\_Single;

dma\_init.DMA\_PeripheralBurst = DMA\_PeripheralBurst\_Single;

DMA\_Init(DMA1\_Stream5, &dma\_init);

DMA\_Cmd(DMA1\_Stream5, ENABLE);

DAC\_Cmd(DAC\_Channel\_1, ENABLE);

DAC\_DMACmd(DAC\_Channel\_1, ENABLE);

}

После компиляции программы-примера и прошивки отладочной платы следует наблюдать за напряжением на выходе PA4 с помощью мультиметра. Прибор должен показать ступенчатое изменение напряжение от 0 В до определенного значения, после чего цикл повторяется. Шаг увеличения одинаков для всего цикла, что можно увидеть по массиву значений для записи в регистр данных.

**Ход работы**

1. Проверить работу программы-примера, скомпилировав ее в одной из сред разработки и выполнив прошивку.

2. Ознакомиться с документацией по DMA для микроконтроллера на отладочной плате.

3. Попробовать изменить определенные параметры в программе-примере.

4. Организовать взаимодействие DMA и другого периферийного устройства в соответствии с индивидуальным заданием.

**Индивидуальные задания**

1. Реализовать асинхронную передачу данных через USART, считывая данные через определенные

промежутки времени посредством DMA.

2. Организовать прием данных через USART с записью в память через DMA. Количество передаваемых

данных известно заранее и равно размеру массива.

3. Продемонстрировать передачу данных через SPI из памяти в циклическом режиме.

4. Продемонстрировать прием данных через SPI с записью в память. Режим записи – циклический.

5. Реализовать запись в память значений, полученных с помощью работы АЦП через определенные

промежутки времени. После записи первых 50 значений начинается новый чикл записи.