**Лабораторная работа № 4**

**Цель работы:** Изучение основных особенностей работы с цифро-аналоговым преобразователем (ЦАП) при программировании для микроконтроллеров (МК) ARM. Разработка программы для генерации синусоиды.

**Приборы и материалы:**

1. Отладочная плата MDR1986VE91T Rev 4.
2. Программатор J-Link ARM.
3. Блок питания 5В, 1.4А.
4. ПК с установленной средой программирования Keil uVision.
5. Осциллограф.
6. Коаксиальный кабель для соединения двух BNC выходов.

**Ход работы**

1. Собрать аппаратную часть по рекомендациям лабораторной работы № 2
2. Открыть проект MDRProject в среде программирования Keil uVision.
3. Подключить к проекту библиотеку DAC, необходимую для работы с ЦАП.

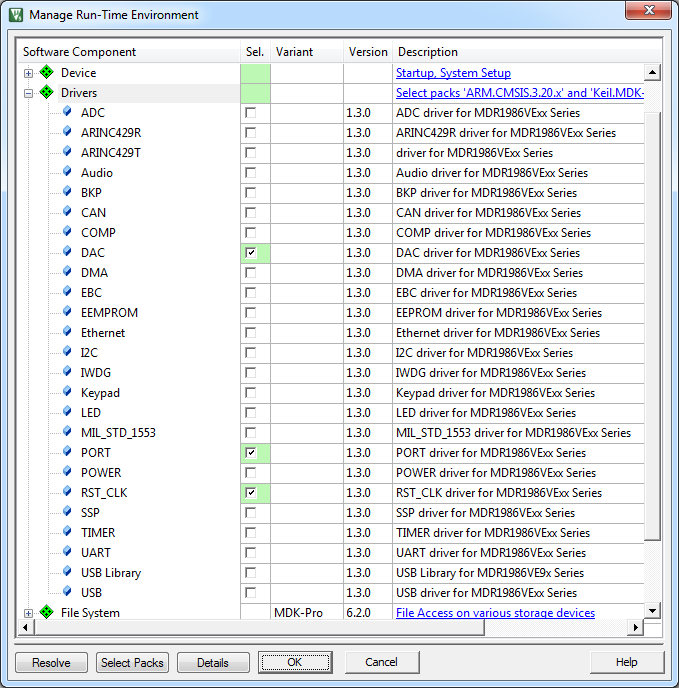


Рисунок 1 – Библиотеки проекта для работы с ЦАП

1. Стереть имеющийся в файле *main.c* исходный код и добавить в начало заголовочные файлы, необходимые в данном проекте:

#include <MDR32F9Qx\_port.h>

#include <MDR32F9Qx\_rst\_clk.h>

#include <MDR32F9Qx\_dac.h>

#include <math.h>

1. Добавить функцию инициализации порта для ЦАП

PORT\_InitTypeDef PORTEInit; //Объявление структуры

void DACPortInit(){

PORT\_StructInit(&PORTEInit); //Загрузка умолчаний

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTE, ENABLE); //Тактование

/\* Настройка DAC1 (стр.11 спецификации) \*/

PORTEInit.PORT\_Pin = PORT\_Pin\_9; //Пин 9

PORTEInit.PORT\_OE = PORT\_OE\_OUT; // Порт E

PORTEInit.PORT\_MODE = PORT\_MODE\_ANALOG; //Режим Аналоговый

PORT\_Init(MDR\_PORTE, &PORTEInit); //Настройка порта

}

1. Добавить функцию инициализации ЦАП

void DACInit(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_DAC, ENABLE); //Тактование

DAC1\_Init(DAC1\_AVCC); //Настройка DAC1 на работу с AVCC

DAC1\_Cmd(ENABLE); //Активация DAC1

}

Функция **DAC1\_Init()** позволяет настроить источник опорного напряжения для ЦАП. Если предполагается использование внешнего источника, необходимо передать в функцию значение **DACn\_REF**, где . В противном случае, для использования напряжения питания МК в качестве опорного, необходимо передать в функцию значение **DACn\_AVCC**.

1. Добавить в файл исходного кода функцию **main()**.

float a;

int main() {

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_RST\_CLK, ENABLE);

DACPortInit();

DACInit();

while(1){

for (a=0; a<360; a+=5)

DAC1\_SetData((sinf(a\*PI/180)\*SCALE + SCALE)/2);

}

}

В приведенном выше фрагменте кода выполняется вызов функции **DAC1\_SetData()** с аргументами, которые вычисляются по функции синуса от равномерно увеличивающегося аргумента. Так как синус находится в пределах от -1 до 1, его необходимо масштабировать в отрезок от 0 до максимального значения. Так как ЦАП 12-битный, максимальное значение равно 0xFFF (12 двоичных единиц). Таким образом, для перевода значений аргумента функции **DAC1\_SetData()** в вольты, необходимо использовать формулу , где *Vcc* является напряжением питания МК (3,3В). С учётом сказанного, необходимо добавить определения констант после подключения библиотек.

#define PI 3.14159265

#define MAX 2 //Volt

#define SCALE (0x7FF \* MAX) / 3.3

1. Скомпилировать исходный код и загрузить его в МК
2. Переключить перемычку **DAC\_OUT\_SEL** в положение **EXT\_CON** (рисунок 1).

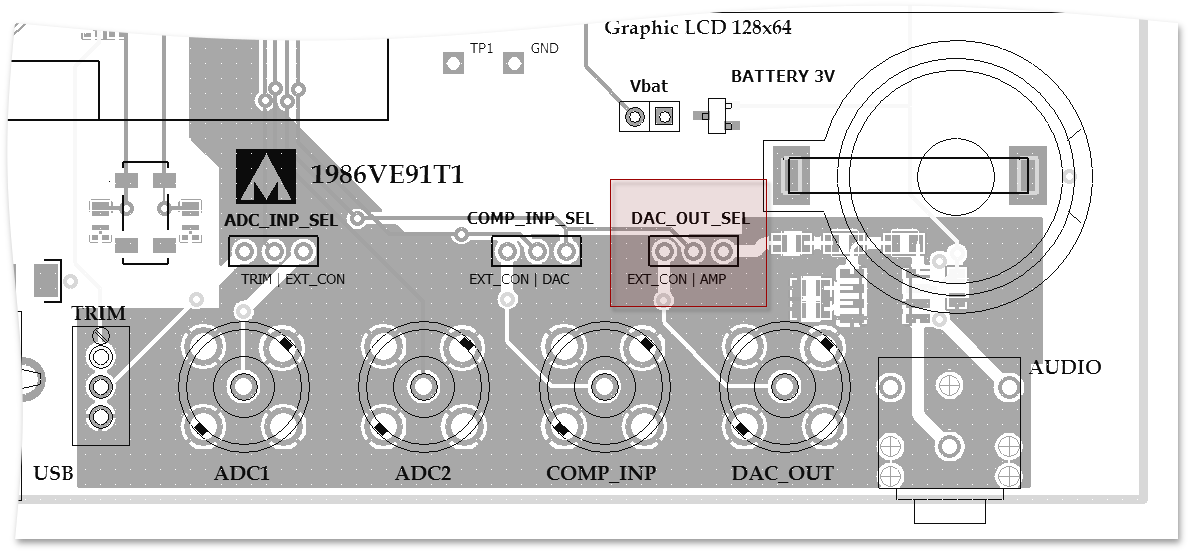


Рисунок 1 - Перемычка DAC\_OUT\_SEL

1. Соединить BNC вывод **DAC\_OUT** со входом осциллографа.
2. Настроить вертикальный и горизонтальный масштаб осциллографа до получения синусоид, аналогичных изображенным на рисунке 2.

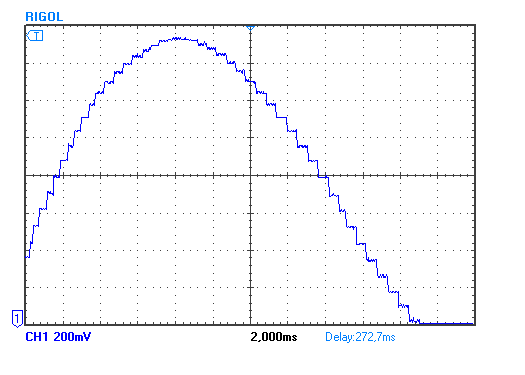
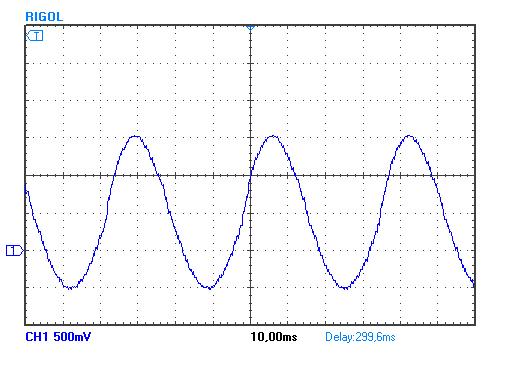


Рисунок 2 – Осциллограммы работы алгоритма

Обратите внимание на ступенчатую структуру синусоиды. Это происходит из-за того, что шаг изменения аргумента синуса 5 градусов, что недостаточно мало для гладкой синусоиды.

**Задание 1**

Вывести на осциллограф симметричную пилообразную функцию (рисунок 3).

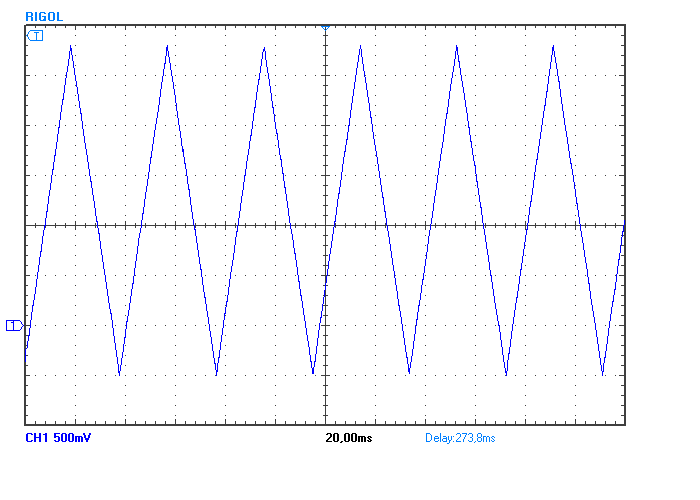


Рисунок 3 – Симметричная пилообразная функция

**Задание 2**

Вывести на осциллограф параболу  в диапазоне . В промежутках между периодичным выводами парабол, поддерживать значение напряжения на уровне 3В. Пример результирующей осциллограммы представлен на рисунке 4.

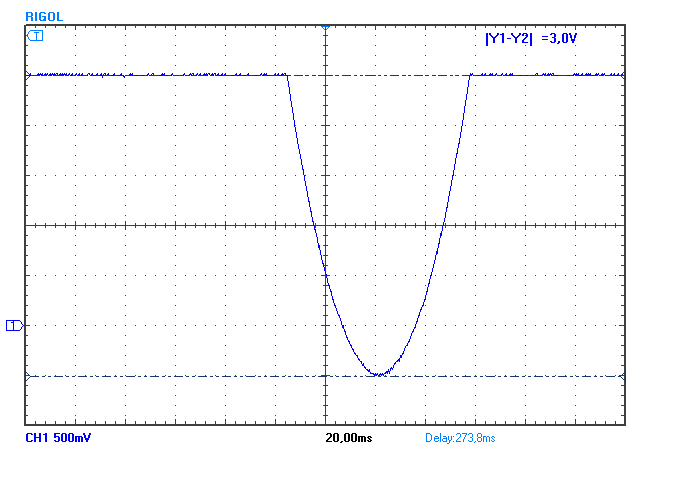


Рисунок 4 – Парабола

Приложение: полный текст программы

#include <MDR32F9Qx\_port.h>

#include <MDR32F9Qx\_rst\_clk.h>

#include <MDR32F9Qx\_dac.h>

#include <math.h>

#define PI 3.14159265

#define MAX 3 //Volt

#define SCALE (0xFFF \* MAX) / 3.3

PORT\_InitTypeDef PORTEInit;

void DACPortInit(){

PORT\_StructInit(&PORTEInit); //Load defaults

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_PORTE, ENABLE);

PORTEInit.PORT\_Pin = PORT\_Pin\_9;

PORTEInit.PORT\_OE = PORT\_OE\_OUT;

PORTEInit.PORT\_MODE = PORT\_MODE\_ANALOG;

PORT\_Init(MDR\_PORTE, &PORTEInit);

}

void DACInit(){

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_DAC, ENABLE);

DAC1\_Init(DAC1\_AVCC);

DAC1\_Cmd(ENABLE);

}

float a;

int i; //4 task 2

int main() {

RST\_CLK\_PCLKcmd(RST\_CLK\_PCLK\_RST\_CLK, ENABLE);

DACPortInit();

DACInit();

while(1){

//Jumper to EXT\_CON!!!

//\* //Base part

for (a=0; a<360; a+=5)

DAC1\_SetData((sinf(a\*PI/180)\*SCALE + SCALE)/2);

//\*/

/\* //Task 1

for (a=0; a<0xFFF; a+=5) DAC1\_SetData(a);

for (a=0xFFF-1; a>1; a-=5) DAC1\_SetData(a);

//\*/

/\* //Task 2

for (a=-0x7FF; a<0; a+=5)

DAC1\_SetData(a\*a/(0x7FF\*0x7FF)\*SCALE);

for (a=0; a<0x7FF; a+=5) DAC1\_SetData(a\*a/(0x7FF\*0x7FF)\*SCALE);

DAC1\_SetData(SCALE);

for (i=0; i<0xFFFF; i++);

//\*/

}

}