**Лабораторная работа № 5**

**Цель работы:** Изучение основных особенностей работы с аналого-цифровым преобразователем (АЦП) при программировании для микроконтроллеров (МК) ARM.

**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. Цифровой вольтметр

**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 ADC, необходимую для работы с АЦП.

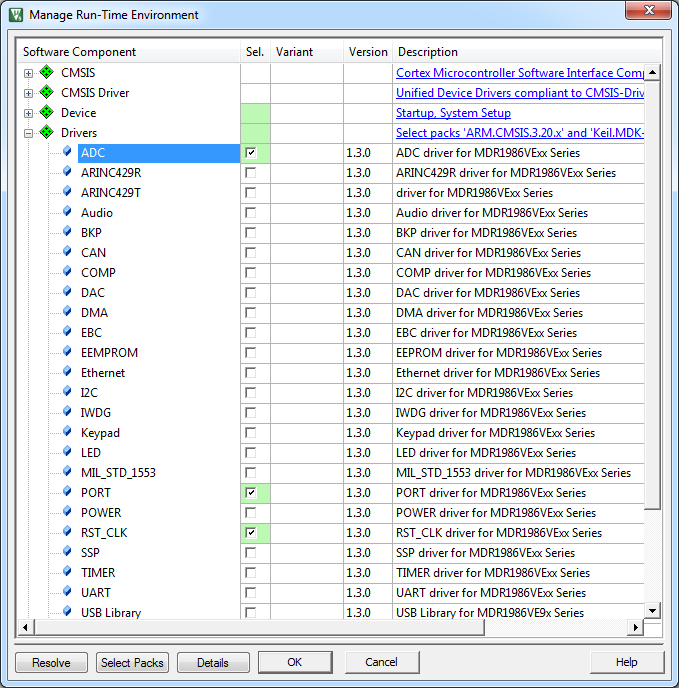


Рисунок 1 – Библиотеки проекта для работы с АЦП

1. Очистить содержимое файла *main.c* и добавить ссылки на подключенные библиотеки:

#include <MDR32F9Qx\_port.h>

#include <MDR32F9Qx\_rst\_clk.h>

#include <MDR32F9Qx\_adc.h>

#include <stdbool.h>

5. Добавить в файл исходного кода функцию **ADCInit()**, в которой произвести инициализацию седьмого канала АЦП1 (на отладочной плате к этому каналу подключено переменное сопротивление).

ADC\_InitTypeDef ADC; //Общая инициализацинная структура подсистемы АЦП

ADCx\_InitTypeDef ADC1; //Инициализацинная структура для АЦП1

void ADCInit(){

//Подача тактования на процессор и АЦП

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

ADC\_StructInit(&ADC); //Заполнение структуры умолч. значениями

ADC\_Init(&ADC); //Инициализация

ADCx\_StructInit(&ADC1);

ADC1.ADC\_ChannelNumber = ADC\_CH\_ADC7; //Выбор седьмого канала

ADC1\_Init(&ADC1);

//Инициализация прерываний АЦП

NVIC\_EnableIRQ(ADC\_IRQn);

NVIC\_SetPriority(ADC\_IRQn, 0);

//Включение прерываний по окончанию преобразования

ADC1\_ITConfig(ADCx\_IT\_END\_OF\_CONVERSION, ENABLE);

//Включение АЦП1

ADC1\_Cmd(ENABLE);

}

6. Добавить в файл исходного кода обработчик прерывания по окончанию аналого-цифрового преобразования.

bool conInProgress; //Флаг «в процессе преобразования»

unsigned int rawResult; //Необработанный результат

unsigned char channel; //Номер канала

float result; //Результат в вольтах

void ADC\_IRQHandler() { //Обработчик прерываний АЦП

//Проверка что причина прерывания соответствует концу преобразования

if(ADC\_GetITStatus(ADC1\_IT\_END\_OF\_CONVERSION)){

rawResult = ADC1\_GetResult(); //Получение результата

channel = (rawResult & 0x1F0000) >> 16; //Сохранение номера канала

rawResult &= 0xFFF; //Удаление номера канала из результата

//Преобразование результата в вольты

result = (float)rawResult / (float)SCALE;

conInProgress = false; //Очистка флага «в процессе преобр-я»

NVIC\_ClearPendingIRQ(ADC\_IRQn); //Очистка флага прерывания

}

}

Как видно из кода, результат содержит в себе не только преобразованное значение напряжения на канале, но и номер этого канала, который нужно убрать из результата. На странице 323 спецификации имеется таблица 301 (рисунок 2), иллюстрирующая это.

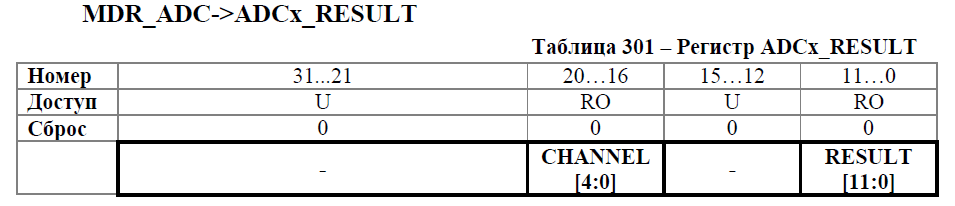


Рисунок 2 ­– Структура регистра с результатом преобразования

С помощью встроенного в Windows калькулятора в режиме программиста, выясняем что битовая маска, зануляющая все биты кроме блока с 20 по 16 имеет вид, отображенный в шестнадцатеричном формате на рисунке 3.

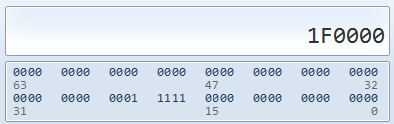


Рисунок 3 – Битовая маска для вычисления номера канала

Конъюнктивно применяя данную битовую маску к результату и смещая его на 16 разрядов вправо (для избавления от 16и значимых нулей в конце), получаем номер канала, который теперь можно занулить (с чистой душой, поскольку ни один бит информации не был потерян). Для этого применяем к результату битовую маску 0xFFF (12 единиц в двоичном представлении), что позволяет занулить оставшуюся часть регистра, не принадлежащую первым 12-и битам результата преобразования.

Далее, необходимо преобразовать полученный результат из неизвестных единиц, пропорциональных напряжению, в вольты. Для этого, результат необходимо поделить на некий коэффициент SCALE, значение которого предстоит определить экспериментальным путём.

7. Последним штрихом является написание функции **main()**, с которой начнётся исполнение программы.

int i; //Счетчик для задержки циклом

int main() {

ADCInit();

while(1){

for(i = 0xFFFF; i > 0; i--); //Задержка циклом (плохой вариант)

if (!conInProgress){ //Не выполняется ли преобразование?

ADC1\_Start(); //Начать преобразование!

conInProgress = true; //Преобразование выполняется

}

}

}

Таким образом, в главном цикле через определенный промежуток времени запускается аналого-цифровое преобразование на шестом канале первого АЦП. По окончании преобразования, вызывается соответствующий обработчик прерывания, в котором результат преобразования заносится в переменную result. Чтобы данный код заработал, не хватает константы SCALE, и чтобы найти такое ее значение, с помощью которого можно получить результат в вольтах, зададим ее равной единице.

#define SCALE 1

8. Подать питание на плату и загрузить программу в микроконтроллер по рекомендациям лабораторной работы №2

9. Поставить перемычку ADC\_INP\_SEL в положение TRIM для подключения переменного сопротивления в качестве источника сигнала для АЦП (рисунок 4).

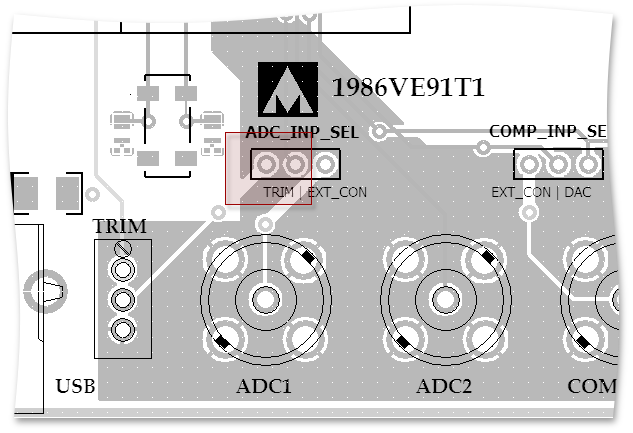


Рисунок 4 – переключение аналогового входа

10. Подключить плюсовой вход вольтметра к среднему контакту перемычки (можно замкнуть перемычку в положение TRIM с помощью щупа вольтметра)

11. Подключить минусовой контакт вольтметра к любому общему проводу отладочной платы (удобно использовать большие крепёжные отверстия по краям платы)

12. Запустить сеанс отладки в программе Keil uVision (Ctrl+F5 или кнопка  на панели инструментов)

13. Добавить переменную **result** в **Watch 1** (правый клик по переменной в коде | *Add ‘result’ to…* | *Watch 1*) и запустить код на исполнение (F5 или кнопка  на панели инструментов)

14. Вращать вал переменного сопротивления, пока напряжение на АЦП не станет равным 2 вольта.

15. Считать из окна **Watch 1** отладчика значение переменной **result** при известном напряжении на АЦП (рисунок 5).

16. Поделить полученное цифровое значение на напряжение, которому оно соответствует.

18. Остановить отладку.

17. Полученный коэффициент вписать в код в качестве константы SCALE.

19. Загрузить код с новым коэффициентом SCALE в МК.

20. Запустить сеанс отладки и убедиться что переменная result отображает значение, совпадающее с показаниями вольтметра.

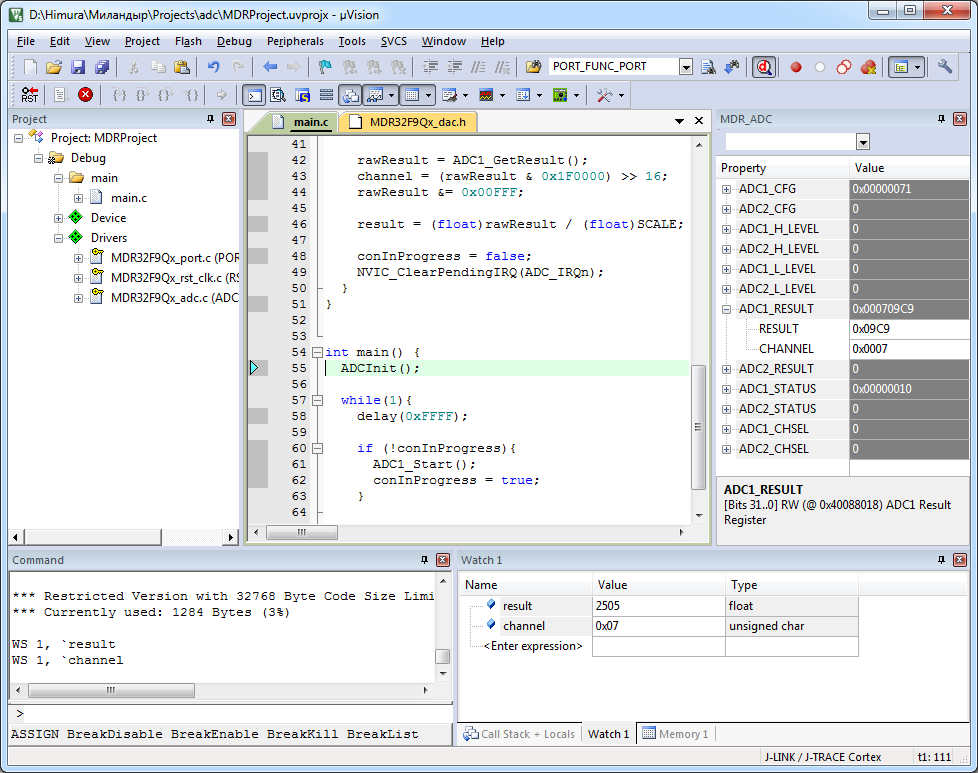


Рисунок 5 – Окно Keil uVision в процессе отладки

В окне среды Keil uVision на рисунке 5 также открыто окно регистров **MDR\_ADC**. Данное окно можно открыть в меню *System Viewer Windows*() на панели инструментов. При анализе содержимого регистра **ADC1\_RESULT**, можно подтвердить правильность его описания (рисунок 2) и даже понять его структуру при отсутствии описания. Отладка является мощнейшим инструментом разработки и помогает увидеть любые ошибки времени выполнения легко и наглядно.

**Задание**

Используя знания, полученные в лабораторной работе № 4, создать цифровой преобразователь сигнала. При входном уровне сигнала (от переменного сопротивления) от 0 до 1 вольта, он должен выдавать сигнал от 2 до 3 вольт аналогичной формы.