

# Микроконтроллеры STM32

Домашнее задание №8

Евгений Зеленин

27 марта 2025 г.

## 1. Постановка задачи

### Условия задачи.

Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Подключите библиотеку шифрования соответствующую имеющемуся у вас микроконтроллеру и compileйте один из представленных примеров. Либо создайте свой пример, опираясь на предоставленные с библиотекой исходники. Целью домашнего задания является: 1. Научиться подключать внешние библиотечные файлы; 2. Получить опыт шифрования данных; 3. Получить опыт работы с предкомпилированными библиотечными файлами. В качестве результата выполнения домашнего задания представьте снимок экрана отображающий в процессе отладки массив с исходными и зашифрованными данными.

## 2. Подключение и настройка библиотеки

Для выполнения этого задания подключим библиотеку с поддержкой fpu к проекту. Для

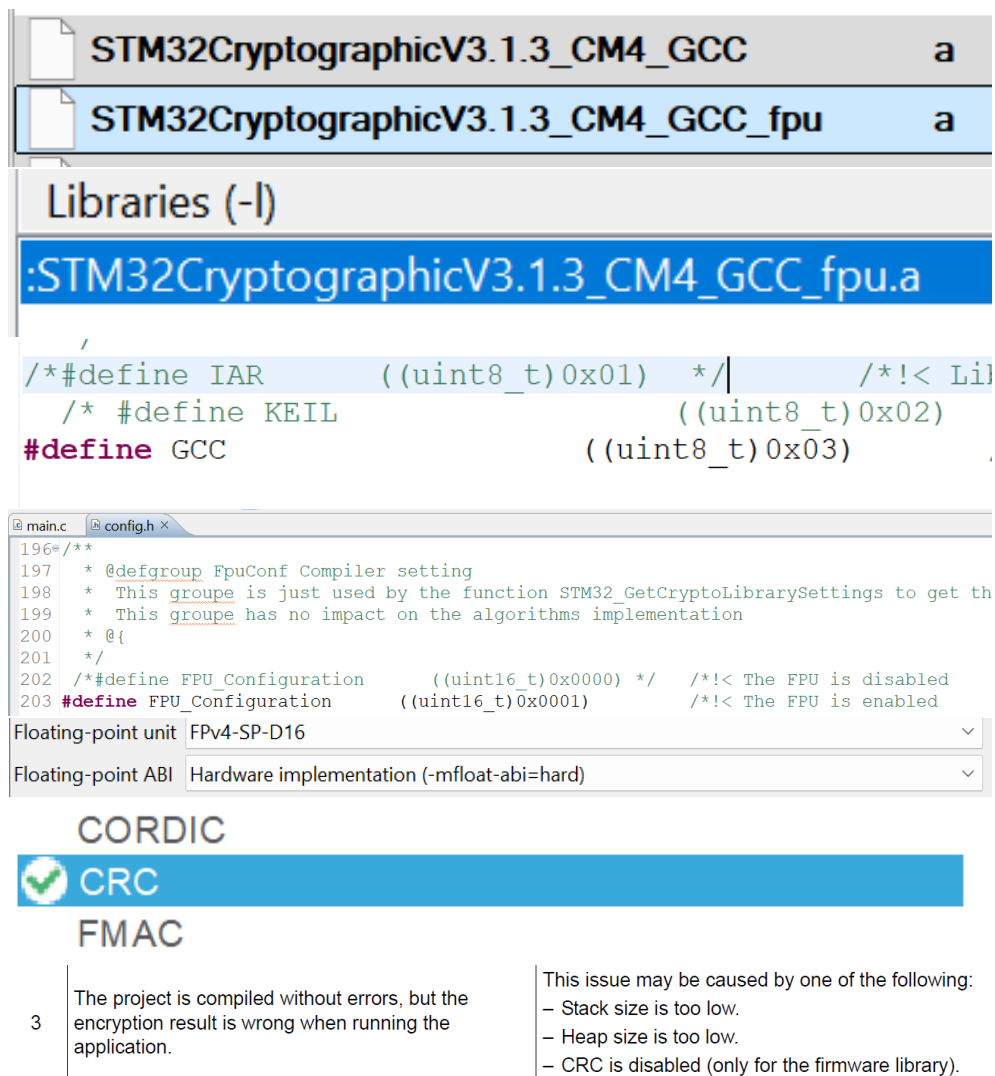


Рис. 1: Подключение и настройка библиотеки

этого скопируем папку STM32\_Cryptographic из раздела с библиотекой под конкретный МК

в наш проект. Далее, произведем настройку в соответствии с UM2388 (рисунок 1).,

Стоит отметить, что для корректной работы библиотеки требуется включить CRC, иначе результат расчета алгоритмов будет ошибочным.

Также, в файле config.h требуется произвести настройку библиотеки, а именно: указать версию X.Y.Z, тип компилятора, поддержку FPU, тип процессора, указать с какой оптимизацией собирается проект и т.д.

### 3. Запуск алгоритмов MD5 и SHA1

Для демонстрации работы посчитаем хэш от строки "EvgeniyZelenin" алгоритмами SHA1 и MD5. Опять же, воспользуемся документацией UM2388, где указан подробный порядок действий по использованию функций библиотеки.

Figure 29. Hash HHH flowcharts

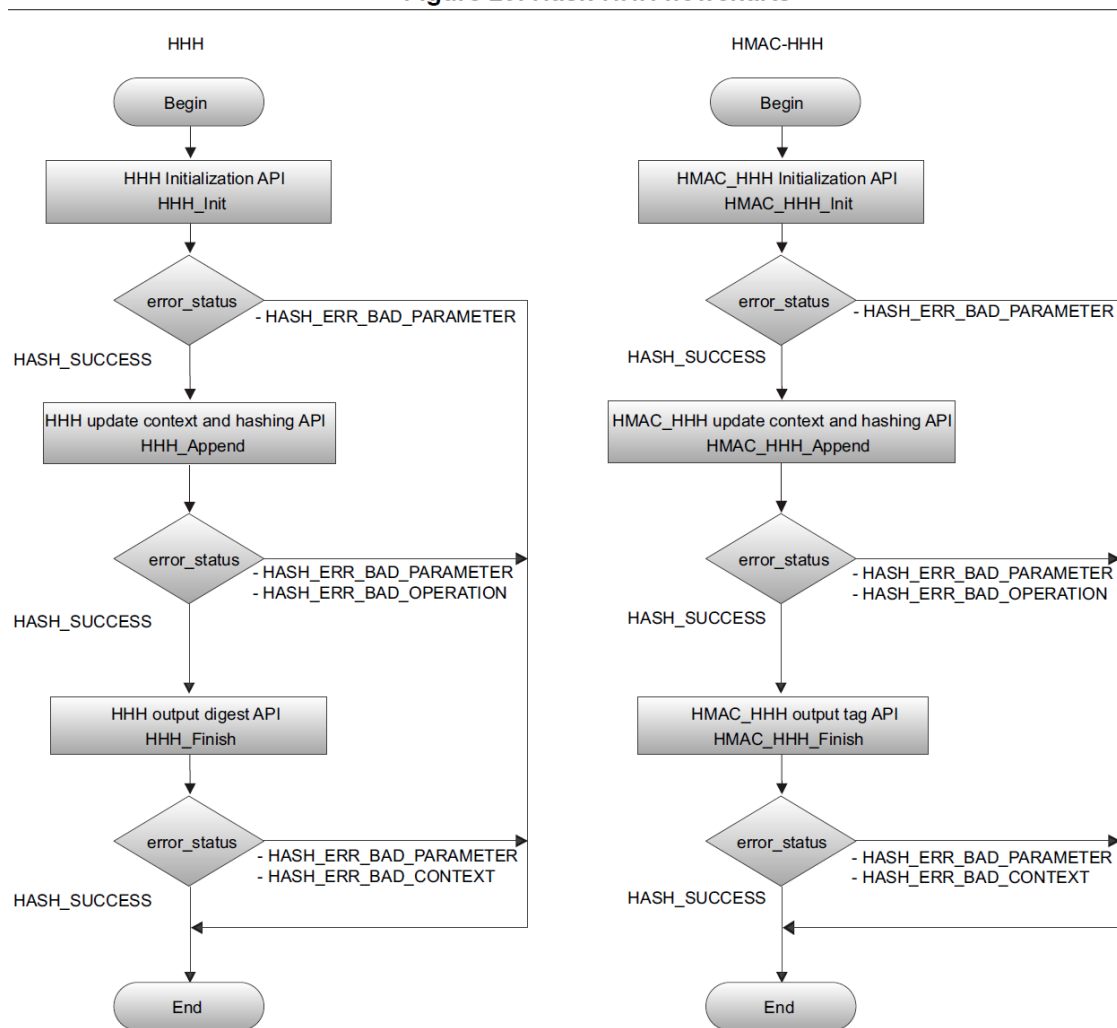


Рис. 2: Алгоритм использования HASH функций

Остается только реализовать эти алгоритмы в коде:

```
/* USER CODE BEGIN WHILE */
```

```

uint8_t input[] = "EvgeniyZelenin";
uint8_t digest[CRL_SHA1_SIZE], digestMD5[CRL_MD5_SIZE];
int32_t outSize, outSizeMD5;
int32_t SHA1_res, MD5_res;

SHA1ctx_stt SHA1ctx_st;
SHA1ctx_st.mTagSize = CRL_SHA1_SIZE;
SHA1ctx_st.mFlags = E_HASH_DEFAULT;
SHA1_res = SHA1_Init(&SHA1ctx_st);
if (SHA1_res == HASH_SUCCESS) {
    SHA1_res = SHA1_Append(&SHA1ctx_st, input, sizeof(input) - 1);
    if (SHA1_res == HASH_SUCCESS) {
        SHA1_res = SHA1_Finish(&SHA1ctx_st, digest, &outSize);
        if (SHA1_res == HASH_SUCCESS) {
            __NOP();
        } else {
            __NOP();
        }
    } else {
        __NOP();
    }
} else {
    __NOP();
}

MD5ctx_stt MD5ctx_st;
MD5ctx_st.mTagSize = CRL_MD5_SIZE;
MD5ctx_st.mFlags = E_HASH_DEFAULT;
MD5_res = MD5_Init(&MD5ctx_st);
if (MD5_res == HASH_SUCCESS) {
    MD5_res = MD5_Append(&MD5ctx_st, input, sizeof(input)-1);
    if (MD5_res == HASH_SUCCESS) {
        MD5_res = MD5_Finish(&MD5ctx_st, digestMD5, &outSizeMD5);
        if (MD5_res == HASH_SUCCESS) {
            __NOP();
        } else {
            __NOP();
        }
    } else {
        __NOP();
    }
} else {
    __NOP();
}
}

```

Функции NOP() здесь использованы для установки точек останова. После выполнения ал-

горитмов, посмотрим какой результат получился, пересчитаем его с использованием online калькулятора для этой же строки и сравним значения.

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */

uint8_t input[] = "EvgeniyZelenin";
uint8_t digest[CRL_SHA1_SIZE], digestMD5[CRL_MD5_SIZE];
int32_t outSize, outSizeMD5;
int32_t SHA1_res, MD5_res;

SHA1ctx_stt SHA1ctx_st;
SHA1ctx_st.mTagSize = CRL_SHA1_SIZE;
SHA1ctx_st.mFlags = E_HASH_DEFAULT;
SHA1_res = SHA1_Init(&SHA1ctx_st);
if (SHA1_res == HASH_SUCCESS) {
    SHA1_res = SHA1_Append(&SHA1ctx_st, input, sizeof(input));
    if (SHA1_res == HASH_SUCCESS) {
        SHA1_res = SHA1_Finish(&SHA1ctx_st, digest, &outSize);
        if (SHA1_res == HASH_SUCCESS) {
            /* Infinite loop */
            /* USER CODE BEGIN WHILE */

            //uint8_t input[] = { 0x40, 0x41 };
            uint8_t input[] = "EvgeniyZelenin";
            uint8_t digest[CRL_SHA1_SIZE], digestMD5[CRL_MD5_SIZE];
            int32_t outSize, outSizeMD5;
            int32_t SHA1_res, MD5_res;

            SHA1ctx_stt SHA1ctx_st;
            SHA1ctx_st.mTagSize = CRL_SHA1_SIZE;
            SHA1ctx_st.mFlags = E_HASH_DEFAULT;
            SHA1_res = SHA1_Init(&SHA1ctx_st);
            if (SHA1_res == HASH_SUCCESS) {
                SHA1_res = SHA1_Append(&SHA1ctx_st, input, sizeof(input));
                if (SHA1_res == HASH_SUCCESS) {
                    SHA1_res = SHA1_Finish(&SHA1ctx_st, digest, &outSize);
                    if (SHA1_res == HASH_SUCCESS) {
                        /* add application taintment in case of SUCCESS */
                        NOP();
                    } else {
                        /* Infinite loop */
                        /* USER CODE BEGIN WHILE */

```

digestMD5	uint8_t [16]	0x20007ee4 (Hex)
digestMD5[0]	uint8_t	0x41 (Hex)
digestMD5[1]	uint8_t	0x2c (Hex)
digestMD5[2]	uint8_t	0x3f (Hex)
digestMD5[3]	uint8_t	0x4b (Hex)
digestMD5[4]	uint8_t	0x90 (Hex)
digestMD5[5]	uint8_t	0x96 (Hex)
digestMD5[6]	uint8_t	0x7e (Hex)
digestMD5[7]	uint8_t	0xf7 (Hex)
digestMD5[8]	uint8_t	0xbd (Hex)
digestMD5[9]	uint8_t	0x82 (Hex)
digestMD5[10]	uint8_t	0xa2 (Hex)
digestMD5[11]	uint8_t	0xcd (Hex)
digestMD5[12]	uint8_t	0xb3 (Hex)
digestMD5[13]	uint8_t	0x92 (Hex)
digestMD5[14]	uint8_t	0x32 (Hex)
digestMD5[15]	uint8_t	0x9e (Hex)

digest	uint8_t [20]	0x20007efc (Hex)
digest[0]	uint8_t	0x27 (Hex)
digest[1]	uint8_t	0xca (Hex)
digest[2]	uint8_t	0x81 (Hex)
digest[3]	uint8_t	0x14 (Hex)
digest[4]	uint8_t	0x93 (Hex)
digest[5]	uint8_t	0x7e (Hex)
digest[6]	uint8_t	0x17 (Hex)
digest[7]	uint8_t	0x95 (Hex)
digest[8]	uint8_t	0xf2 (Hex)
digest[9]	uint8_t	0xd4 (Hex)
digest[10]	uint8_t	0x3b (Hex)
digest[11]	uint8_t	0x5 (Hex)
digest[12]	uint8_t	0x4a (Hex)
digest[13]	uint8_t	0xec (Hex)
digest[14]	uint8_t	0xb9 (Hex)
digest[15]	uint8_t	0x22 (Hex)
digest[16]	uint8_t	0x80 (Hex)
digest[17]	uint8_t	0x33 (Hex)
digest[18]	uint8_t	0xe8 (Hex)
digest[19]	uint8_t	0xea (Hex)

EvgeniyZelenin

Encode

## SHA1 encoded string

27ca8114937e1795f2d43b054aecb9228033e8ea

## Your last 10 encodings x

Algorithm	String	Hash
sha1	EvgeniyZelenin	27ca8114937e1795f2d43b054aecb9228033e8ea
md5	EvgeniyZelenin	412c3f4b90967ef7bd82a2cdb392329e

Рис. 3: Сравнение хэшей. Сверху-вниз: MD5, SHA1, online-калькулятор

Как видно из рисунка 3, результаты вычислений совпадают. В материалах к занятию приложен проект и скриншоты.

#### 4. **Дополнительные материалы**

Материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №08