Микроконтроллеры STM32

Курсовая работа

Евгений Зеленин

7 апреля 2025 г.

1. Постановка задачи

Условия задачи.

В качестве курсового проекта вам предстоит создать свой собственный загрузчик. Создаваемое вами программное обеспечение должно уметь выполнять следующие действия: Необходимо прочитать сохраненные качестве констант строки HEX файла. Достаточно использовать 5-6 строк с разнообразными данными. Написать функцию проверяющую корректность CRC кода считанной строки, анализирующую структуру записи HEX файла и помещает данные в память микроконтроллера. В качестве результата работы представить файл main.c проекта, снимок экрана демонстрирующий фрагмент flash памяти с записанными в нее данными.

2. Структура памяти stm32g431

Для этого проекта будем использовать отладочную плату на базе stm32g431cbt6. Обратимся к спецификации на микроконтроллер и выясним как организована память данного МК (см. рисунок1).

Table 7. Flash module - 512/256/128 KB dual bank organization (64 bits read width)

Flash area		Flash memory addresses	Size (bytes)	Name
Main memory (512/256/128 KB)	Bank 1(1) (256/128/64 KB)	0x0800 0000 - 0x0800 07FF	2 K	Page 0
		0x0800 0800 - 0x0800 0FFF	2 K	Page 1
		0x0800 1000 - 0x0800 17FF	2 K	Page 2
		0x0800 1800 - 0x0800 1FFF	2 K	Page 3
		⋮	⋮	⋮
		0x0803 F800 - 0x0803 FFFF	2 K	Page 127
	Bank 2(1) (256/128/64 KB)	0x0804 0000 - 0x0804 07FF	2 K	Page 0
		0x0804 0800 - 0x0804 0FFF	2 K	Page 1
		0x0804 1000 - 0x0804 17FF	2 K	Page 2
		0x0804 1800 - 0x0804 1FFF	2 K	Page 3
		⋮	⋮	⋮
		0x0807 F800 - 0x0807 FFFF	2 K	Page 127
Information block	Bank 1	0x1FFF 0000 - 0x1FFF 6FFF	28 K	System memory
	Bank 2	0x1FFF 8000 - 0x1FFF EFFF	28 K	
	Bank 1	0x1FFF 7000 - 0x1FFF 73FF	1 K	OTP area
	Bank 1	0x1FFF 7800 - 0x1FFF 782F	48	Option bytes
	Bank 2	0x1FFF F800 - 0x1FFF F82F	48	

1. For 256KB devices: from page 0 to page 63
   For 128KB devices: from page 0 to page 31

Рис. 1: Организация памяти stm32g431

Как видно из рисунка 1, память организовано постранично, размер одной страницы - 2K. У нашего микроконтроллера 128K памяти. Тогда первый банк памяти начинается с адреса 0x0800 0000, а второй банк памяти - с адреса 0x0801 0000. Особенностью данного МК является то, что мы можем закрыть от чтения/записи один из банков памяти или ее часть, а другую

оставить открытой. Т.е. можно сделать защищенный загрузчик, который находится в bank1, а прошивку устройства хранить в bank2 или наоборот.

Также, такая организация памяти позволяет реализовать режим dualboot. С помощью битов BFB2 и nSWAP_BANK (в регистре FLASH_OPTCR) можно выбирать с какого из банков памяти будет запускаться микроконтроллер. Этот режим можно применить для фонового обновления ПО микроконтроллера.

Для того, чтобы сгенерировать hex файл с нужной адресацией, поменяем адресацию в .ld скрипте и настроим сохранение прошивки в hex. После чего, скопируем первые и последние строки из hex файла и вернем адресацию обратно.

```
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 32K
  FLASH    (rx)    : ORIGIN = 0x8000000,   LENGTH = 64K
  /*FLASH    (rx)    : ORIGIN = 0x8010000,   LENGTH = 64K*/
}
```

Рис. 2: Формирование hex файла

Теперь, для демонстрации записи во флеш нам достаточно разместить несколько строк в формате hex в виде массива символов в коде, а затем обработать эти данные и разместить по соответствующим адресам во flash.

```
50 const char *data[] = {
51         ":020000040801F1",
52         ":1000000000800020C10301083D03010845030108E9",
53         ":100010004D030108550301085D03010800000000BD",
54         ":1000200000000000000000000000006503010850F",
55         ":1000300073030108000000008103010088F03010819",
56         ":1000400011040108110401081104010811040108 38",
57         ":1000500011040108110401081104010811040108 28",
58         ":1000600011040108110401081104010811040108 18",
59         ":1000700011040108110401081104010811040108 08",
60         ":1000800011040108000000001104010811040108 16",
61         ":1000900011040108110401081104010811040108E8",
62         ":04000005080103C12A",
63         ":00000001FF" };
```

Рис. 3: Набор hex строк

Для успешной записи нам необходима строка с началом адреса (код записи 04). Любая строка с данными для записи (код 00), а так же строка обозначающая конец файла (код 01). Строки с другими кодами будут игнорироваться.

Также, стоит отметить, что в нашем случае мы можем формировать файл прошивки как нам удобно, поэтому в целях более эффективной организации памяти, сделаем допущение что объем данных в одной строке составляет максимум 16 байт.

3. Логика работы

В учебных целях, нам достаточно лишь показать возможность обработки данных в hex формате и их записи по соответствующим адресам памяти.

После запуска МК сразу же начинается обработка массива с hex строками. Сначала для каждой строки проверяется контрольная сумма. Затем выделяются старшие два байта адреса и записываются в соответствующее поле структуры.

```
  typedef struct {
 uint8_t data[16];
 uint8_t type;
 uint8_t size;
 uint16_t addrh;
 uint16_t addrl;
} my_record;
```

После того как нам становится известны старшие два байта адреса, можно приступить к стиранию flash памяти. В демонстрационных целях нам достаточно стереть одну страницу памяти.

Из последующих строк выделяется младшая часть адреса и точно также записывается в соответствующее поле. После чего в структуру заносится информация о длине данных, а также сами данные.

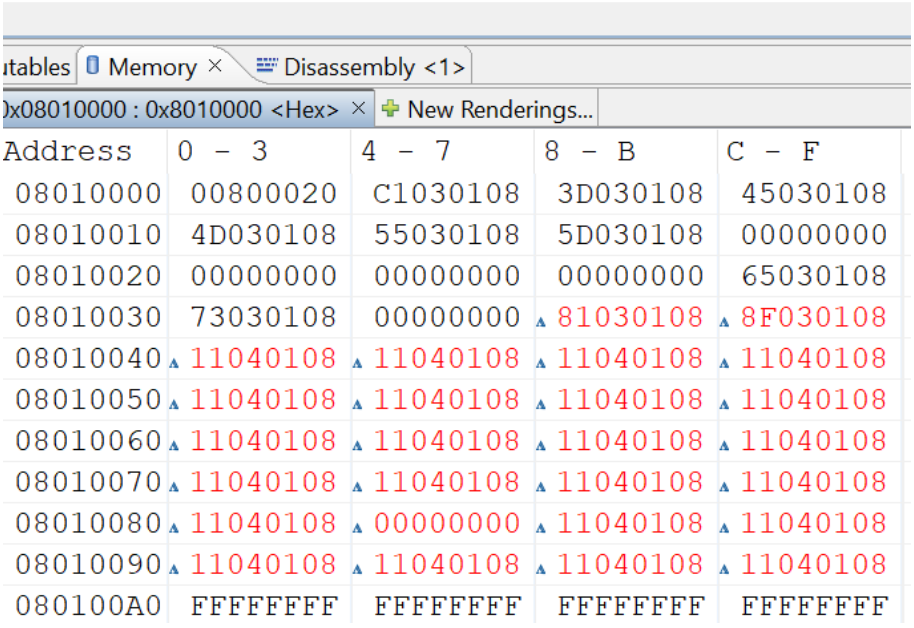
Когда структура сформирована, начинается процесс записи во флеш память двойными словами (doubleword).

Процесс обработки и записи продолжается до тех пор, пока не встретится запись с кодом "01". После чего обработка прошивки прекращается.

Скриншот с образцами данных показан на рисунке 4.

Видео демонстрация процесса записи, а также скриншоты находятся в дополнительных материалах.

```
":020000040801F1",
":1000000000800020C10301083D03010845030108E9",
":100010004D030108550301085D03010800000000BD",
":10002000000000000000000000000650301085F",
":1000300073030108000000000810301088F03010819",
":10004000110401081104010811040108110401080838",
":10005000110401081104010811040108110401080828",
":10006000110401081104010811040108110401080818",
":10007000110401081104010811040108110401080808",
":10008000110401080000000011040108110401080816",
":10009000110401081104010811040108110401080E8",
":04000005080103C12A",
":00000001FF" };
```



Address	0 - 3	4 - 7	8 - B	C - F
08010000	00800020	C1030108	3D030108	45030108
08010010	4D030108	55030108	5D030108	00000000
08010020	00000000	00000000	00000000	65030108
08010030	73030108	00000000	81030108	8F030108
08010040	11040108	11040108	11040108	11040108
08010050	11040108	11040108	11040108	11040108
08010060	11040108	11040108	11040108	11040108
08010070	11040108	11040108	11040108	11040108
08010080	11040108	00000000	11040108	11040108
08010090	11040108	11040108	11040108	11040108
080100A0	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF

Рис. 4: Сравнение данных в массиве и flash памяти

4. Дополнительные материалы

Материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к Курсовому