Осциллограф С8-54

Техническое описание программного обеспечения

ТОПО

Минск, ОАО МНИПИ 2017

Оглавление

[Задействованные аппаратные ресурсы 3](#_Toc474836897)

[Загрузчик 6](#_Toc474836898)

[Настройки 6](#_Toc474836899)

[Создание прошивки 8](#_Toc474836900)

[Хранение серийного номера и номера версии программного обеспечения 9](#_Toc474836901)

[Сохранение настроек 10](#_Toc474836902)

[Сохранение данных в ППЗУ 11](#_Toc474836903)

[Замечания 12](#_Toc474836904)

[Вывод данных 12](#_Toc474836905)

# Задействованные аппаратные ресурсы

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Выводы | Регистры |
| TIM2 | Отсчёт тиков и миллисекунд |  |  |
| TIM3 | Отсчёт временных промежутков для таймеров |  |  |
| TIM5 | Отсчёт миллисекунд |  |  |
| TIM6 | Отсчёт миллисекунд |  |  |
|  |  |  |  |
|  | **Динамик** |  |  |
| DMA1 |  |  |  |
| TIM7 |  |  |  |
| DAC1 |  |  | PA4 |
|  |  |  |  |
|  | **Панель** |  |  |
|  | Программный NSS | 56 | PG0 |
| SPI1 | SCK | 41 | PA5 |
| SPI1 | MISO | 42 | PA6 |
| SPI1 | MOSI | 135 | PB5 |
|  |  |  |  |
|  | **RAM, FPGA, дисплей** |  |  |
| FMC | NE1 | 123 | PD7 |
| FMC | NL1 | 137 | PB7 |
| FMC | NOE/RD | 118 | PD4 |
| FMC | NWE/WR | 119 | PD5 |
| FMC | D0 | 85 | PD14 |
| FMC | D1 | 86 | PD15 |
| FMC | D2 | 114 | PD0 |
| FMC | D3 | 115 | PD1 |
| FMC | D4 | 58 | PE7 |
| FMC | D5 | 59 | PE8 |
| FMC | D6 | 60 | PE9 |
| FMC | D7 | 63 | PE10 |
| FMC | D8 | 64 | PE11 |
| FMC | D9 | 65 | PE12 |
| FMC | D10 | 66 | PE13 |
| FMC | D11 | 67 | PE14 |
| FMC | D12 | 68 | PE15 |
| FMC | D13 | 77 | PD8 |
| FMC | D14 | 78 | PD9 |
| FMC | D15 | 79 | PD10 |
| FMC | A0 | 10 | PF0 |
| FMC | A1 | 11 | PF1 |
| FMC | A2 | 12 | PF2 |
| FMC | A3 | 13 | PF3 |
| FMC | A4 | 14 | PF4 |
| FMC | A5 | 15 | PF5 |
| FMC | A6 | 50 | PF12 |
| FMC | A7 | 53 | PF13 |
| FMC | A8 | 54 | PF14 |
| FMC | A9 | 55 | PF15 |
| FMC | A10 | 56 | PG0 |
| FMC | A11 | 57 | PG1 |
| FMC | A12 | 87 | PG2 |
| FMC | A13 | 88 | PG3 |
| FMC | A14 | 89 | PG4 |
| FMC | A15 | 90 | PG5 |
| FMC | A16 | 80 | PD11 |
| FMC | A17 | 81 | PD12 |
| FMC | A18 | 82 | PD13 |
| FMC | A19 | 2 | PE3 |
| FMC | A20 | 3 | PE4 |
|  |  |  |  |
|  | **Флешка** |  |  |
| OTG FS | VBUS | 101 | PA9 |
| OTG FS | D- | 103 | PA11 |
| OTG FS | D+ | 104 | PA12 |
|  |  |  |  |
|  | **USB** |  |  |
| OTG HS | DM | 75 | PB14 |
| OTG HS | DP | 76 | PB15 |
|  |  |  |  |
|  | **Ethernet** |  |  |
| ETH | MDIO | 36 | PA2 |
| ETH | MDC | 27 | PC1 |
| ETH | CRS | 34 | PA0 |
| ETH | COL | 37 | PA3 |
| ETH | RX CLK | 35 | PA1 |
| ETH | RX ER | 69 | PB10 |
| ETH | RX DV | 43 | PA7 |
| ETH | RXD0 | 44 | PC4 |
| ETH | RXD1 | 45 | PC5 |
| ETH | RXD2 | 46 | PB0 |
| ETH | RXD3 | 47 | PB1 |
| ETH | TX CLK | 29 | PC3 |
| ETH | TX EN | 70 | PB11 |
| ETH | TXD0 | 128 | PG13 |
| ETH | TXD1 | 129 | PG14 |
| ETH | TXD2 | 28 | PC2 |
| ETH | TXD3 | 139 | PB8 |
|  |  |  |  |
|  | **Прерывания IRQ** |  |  |
|  | DMA1\_Stream5\_IRQn – динамик |  |  |
|  | OTG\_FS\_IRQn – флешка |  |  |
|  | OTG\_HS\_IRQn – UBS |  |  |

# **Загрузчик**

Основная программа начинается с адреса 0x08020000 и занимает сектора 5, 6 и 7 размерами по 128 кб.

В начальных секторах находится загрузчик. Его функции:

- при включении загрузчик проверяет наличие подключённой флешки. Если флешка не обнаружена, происходит передача управления на адрес 0x08020000. Если флешка обнаружена:

- происходит поиск в корневом каталоге файла S8-54.bin. Если файл обнаружен, выводится запрос на обновление. В случае выбора варианта «НЕТ» происходит передача управления на адрес 0x08020000. Если выбрано «ДА»:

- стираются сектора 5, 6 и 7; затем в них записывается содержимое файла S8-53.bin, начиная с адреса 0x08020000. Затем происходит переход на адрес 0x08020000.

## Настройки

Проект S8-54

1. Target

Read/Only Memory Areas – on-chip – IROM1 : 0x8000000 0x100000

Code Generation ARM Compiler : Use default compiler version

Read/Write Memory Areas – on-chip – IRAM1 : 0x20000000 0x30000

1. User

After Build/Rebuild – Run#1 :

fromelf –bin –output.\s8-54.bin .\objects\s8-54.afx

1. C/C++

Preprocessor Symbols – Define : USE\_HAL\_DRIVER, STM32F437xx

Language/Code Generation – Optimization : Level 2(-O2)

Language/Code Generation – One ELF Section per Function

Language/Code Generation – Warnings : All Warnings

No Auto Includes

C99 Mode

1. Linker

Report ‘might fail’ Conditions as Errors

R/O Base : 0x08000000

R/W Base : 0x20000000

Scatter File : .\S8-54.sct

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

; \*\*\* Scatter-Loading Description File generated by uVision \*\*\*

; \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LR\_IROM1 0x08020000 0x00100000 { ; load region size\_region

ER\_IROM1 0x08020000 0x00100000 { ; load address = execution address

\*.o (RESET, +First)

\*(InRoot$$Sections)

.ANY (+RO)

}

RW\_IRAM1 0x20000000 0x00030000 { ; RW data

.ANY (+RW +ZI)

}

RW\_IRAM2 0x10000000 0x00010000 { ; CCM data

.ANY (CCM\_DATA) ;FOO ccmFoo[1024] \_\_attribute\_\_ ((section("CCM\_DATA")));

startup\_stm32f437xx.o (STACK)

;startup\_stm32f437xx.o (HEAP)

}

}

1. Debug

CPU DLL : SARMCM3.DLL Parameter : -REMAP –MPU

Dialog DLL : DCM.DLL Parameter : -pCM4

Driver DLL : SARMCM3.DLL Parameter : -MPU

Dialog DLL : TCM.DLL Parameter : -pCM4

Проект BootloaderS8-54

## Создание прошивки

Для того, чтоб получить итоговую прошивку, в её начало нужно записать серийный номер

# Хранение серийного номера и номера версии программного обеспечения

Серийный номер и номер версии программного обеспечения хранятся в секторе 3 размером 16 кБайт по следующим адресам.

0x08010000 – серийный номер в виде НОМЕР (пробел) ГОД, где:

НОМЕР – uint8 (в будущем, возможно, uint16) , где хранится номер прибора;

ГОД – uint32 с годом выпуска.

0x08010010 – версия программного обеспечения в uint32.

Запись этих значений производится через меню ОТЛАДКА-ДАННЫЕ при залитой прошивке.

Также при обновлении через флешку загрузчик перезаписывает версию на актуальную. Актуальная версия прошивки хранится в её первых десяти байтах в виде символьной строки.

# Сохранение настроек

Настройки сохраняются в отдельном секторе. Принцип сохранения/загрузки описан в модуле FLASH.c.

Настройки бывают двух типов – сбрасываемые и несбрасываемые. Сбрасываемые – это те, которые сбрасываются после нажатия «СЕРВИС-Сброс». Несбрасываемые сохраняются после такой манипуляции. К несбрасываемым относятся такие настройка, как заводской номер, версия ПО, настройки входных каналов.

Вначале несбрасываемые настройки хранились в одной структуре со сбрасываемыми, однако затем было решено их разделить – для уменьшения вероятности потери несбрасываемых настроек. Сбрасываемые остались в секторе размером 128кБ, а несбрасываемые перемещены в первый сектор размером 64 кБ. Принцип хранения такой. Сектор разделён на участки, кратные 16кБ (по 512, 256, или 128 байт). В первом байте каждого такого участка хранится число, определяющее кратность учкастка относительно 16 (для 512 байт – будет записано 32, для 256 – 16, для 128 – 8). Затем – собственно настройки.

Существенная особенность несбрасываемых настроек – их количество может измениться со временем. Поэтому, чтобы не утерять те настройки, которые уже были, новые настройки можно дописывать только в конец структуры. Таким образом, предыдущие настройки будут сохранены.

# Хранение данных в ППЗУ

В секторе ADDR\_DATA\_DATA хранится массив структур вида

typedef struct

{

uint lastRecord;

uint addr[MAX\_NUM\_SAVED\_WAVES]

} ArrayDatas.

В lastRecord хранится адрес, по которому прозведена последняя запись. Если lastRecord равен 0, то массив не соответствует текущему состоянию и нужно смотреть следующий. Значение lastRecord == 0xFFFFFFFF (MAX\_UINT) может быть у первого элемента массива, если сохранение данных ещё не производилось.

addr[i] – адрес, по которому записаны данные соответствующего номера в следующем порядке : DataSettings, данные первого канала (если включён), данные второго канала (если включён). Если addr[i] == 0 или addr[i] == MAX\_UINT – данных нет.

Алгоритм сохранения.

1. Проверяем значение lastRecord у нулевого элемента в массиве ADDR\_DATA\_DATA. Если оно равно 0xFFFFFFFF, переходим к процедуре собственно сохранения данных – шаг 4.
2. Перебираем по порядку элементы массива, пока не находим элемент с lastRecord != 0. Это действующая структура.
3. Записываем в addr[i] 0, что равноценно стиранию данных.
4. Определяем, достаточно ли места в ППЗУ для сохранения данных. Для этого пользуемся размером DataSettings, количеством байт в канале, также значением lastRecord и адресом за последним доступным байтом ППЗУ (ADDR\_DATA\_6 + 128 \* 1024). Если места достаточно, переходим к шагу 6.
5. Уплотняем данные. Для этого:
   1. Заводим в ОЗУ структуру ArrayDatas и копируем в неё действующую.
   2. i = 0; // В этой переменной хранится номер сектора для хранения данных (ADDR\_DATA\_0 … ADDR\_DATA\_6).
   3. if (i > 6) – переход к 5.
   4. Переписываем в ОЗУ

# Замечания

## Вывод данных

На экран выводятся точки со значениями от MIN\_VALUE до MAX\_VALUE. MIN\_VALUE – нижняя граница сетки, MAX\_VALUE – верхняя граница сетки. Выходящие за эти границы значения приводятся к границам и тоже выводятся как крайние точки. Если value == 0, то в этом месте данных нет, выводить ничего не надо. Это бывает, когда находимся в поточечном режиме – в начале экрана данные есть, а с какой-то позиции нету.