Микроконтроллеры STM32

Домашнее задание №3

Евгений Зеленин

6 марта 2025 г.

1. Постановка задачи

Условия задачи. *Создайте проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. Определите структуру данных с несколькими полями различного типа. Напишите программу, которая помещает данную структуру во Flash память микроконтроллера по нажатию на кнопку. Целью домашней работы является получение учащимися опыта по работе с flash памятью микроконтроллера, сохранению и чтению смешанных структур данных. В качестве результата выполнения ДЗ представьте файл main.c + снимок экрана демонстрирующий дамп flash памяти микроконтроллера с сохраненной структурой данных.*

2. Запись во встроенну flash память

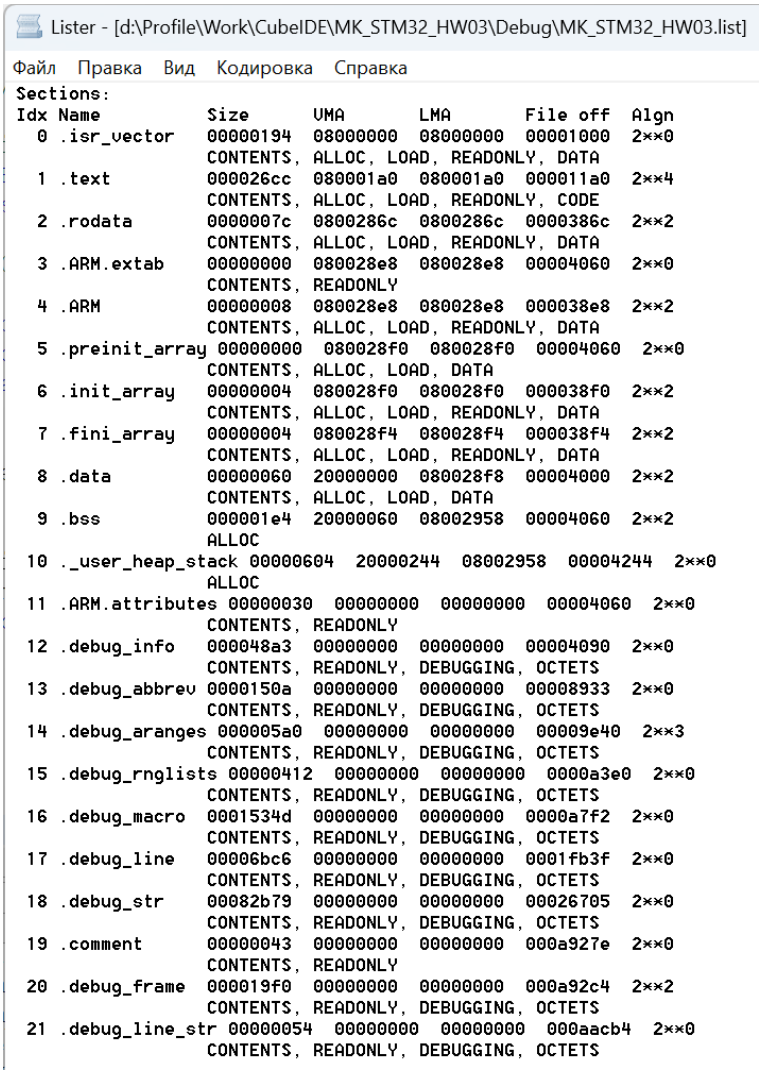
Сначала, определим в какую область памяти мы можем записать данные. Для этого, обратимся к datasheet на микроконтроллер и посмотрим на схему организации памяти (рисунок 1). Как видно из рисунка, память организована по секторам. Минимальная единица для стирания - сектор.

Table 5. Flash module organization (STM32F401xB/C and STM32F401xD/E)

Block	Name	Block base addresses	Size
Main memory	Sector 0	0x0800 0000 - 0x0800 3FFF	16 Kbytes
	Sector 1	0x0800 4000 - 0x0800 7FFF	16 Kbytes
	Sector 2	0x0800 8000 - 0x0800 BFFF	16 Kbytes
	Sector 3	0x0800 C000 - 0x0800 FFFF	16 Kbytes
	Sector 4	0x0801 0000 - 0x0801 FFFF	64 Kbytes
	Sector 5	0x0802 0000 - 0x0803 FFFF	128 Kbytes
	Sector 6	0x0804 0000 - 0x0805 FFFF	128 Kbytes
	Sector 7	0x0806 0000 - 0x0807 FFFF	128 Kbytes
System memory		0x1FFF 0000 - 0x1FFF 77FF	30 Kbytes
OTP area		0x1FFF 7800 - 0x1FFF 7A0F	528 bytes
Option bytes		0x1FFF C000 - 0x1FFF C00F	16 bytes

Рис. 1: Организация памяти STM32F401CCU6

Теперь посмотрим на адреса секций данных нашей прошивки в файле .list.



Lister - [d:\Profile\Work\CubeIDE\MK_STM32_HW03\Debug\MK_STM32_HW03.list]

Файл Правка Вид Кодировка Справка

```
Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .isr_vector   00000194  08000000  08000000  00001000  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  1 .text         000026cc  080001a0  080001a0  000011a0  2**4
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .rodata       0000007c  0800286c  0800286c  0000386c  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  3 .ARM.extab    00000000  080028e8  080028e8  00004060  2**0
                  CONTENTS, READONLY
  4 .ARM          00000008  080028e8  080028e8  000038e8  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  5 .preinit_array 00000000  080028f0  080028f0  00004060  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  6 .init_array   00000004  080028f0  080028f0  000038f0  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  7 .fini_array   00000004  080028f4  080028f4  000038f4  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, DATA
  8 .data         00000060  20000000  080028f8  00004000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  9 .bss          000001e4  20000060  08002958  00004060  2**2
                  ALLOC
 10 ._user_heap_stack 00000604  20000244  08002958  00004244  2**0
                  ALLOC
 11 .ARM.attributes 00000030  00000000  00000000  00004060  2**0
                  CONTENTS, READONLY
 12 .debug_info   000048a3  00000000  00000000  00004090  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 13 .debug_abbrev 0000150a  00000000  00000000  00008933  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 14 .debug_aranges 000005a0  00000000  00000000  00009e40  2**3
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 15 .debug_rnglists 00000412  00000000  00000000  0000a3e0  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 16 .debug_macro  0001534d  00000000  00000000  0000a7f2  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 17 .debug_line   00006bc6  00000000  00000000  0001fb3f  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 18 .debug_str    00082b79  00000000  00000000  00026705  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 19 .comment      00000043  00000000  00000000  000a927e  2**0
                  CONTENTS, READONLY
 20 .debug_frame  000019f0  00000000  00000000  000a92c4  2**2
                  CONTENTS, READONLY, DEBUGGING, OCTETS
 21 .debug_line_str 00000054  00000000  00000000  000aacb4  2**0
                  CONTENTS, READONLY, DEBUGGING, OCTETS
```

Рис. 2: Адресация секций прошивки

Как видно из рисунка, данными занят только первый сектор, соответственно, остальные свободны.

Возьмем сектор 5 для записи в него структуры данных.

Для этого сформируем структуру с различными полями:

```
  typedef struct {
 uint8_t index;
 uint8_t start;
 uint8_t end;

 uint8_t flag1 :1;
 uint8_t flag2 :1;
 uint8_t flag3 :1;
 uint8_t flag4 :1;
 uint8_t flag5 :1;
 uint8_t flag6 :1;
 uint8_t flag7 :1;
 uint8_t flag8 :1;

 float my_float;
 char text[56];

} mystruct;
```

Для побайтовой записи структуры в память, будем использовать объединение.

```
typedef union {
 mystruct data;
 uint8_t bytes[64];
} my_union;
```

Заполним структуру произвольными данными:

```
my_data.data.start = 3;
 my_data.data.end = 60;
 my_data.data.flag1 = 1;
 my_data.data.flag2 = 1;
 my_data.data.flag3 = 0;
 my_data.data.flag4 = 0;
 my_data.data.flag5 = 0;
 my_data.data.flag6 = 0;
 my_data.data.flag7 = 1;
 my_data.data.flag8 = 1;
 my_data.data.my_float = 3.45;
 sprintf(my_data.data.text,
   "Hello world! T e s t  w r i t e  t o  f l a s h  ==  ok");
```

И запишем во flash память по нажатию кнопки по адресу 0x08020000:

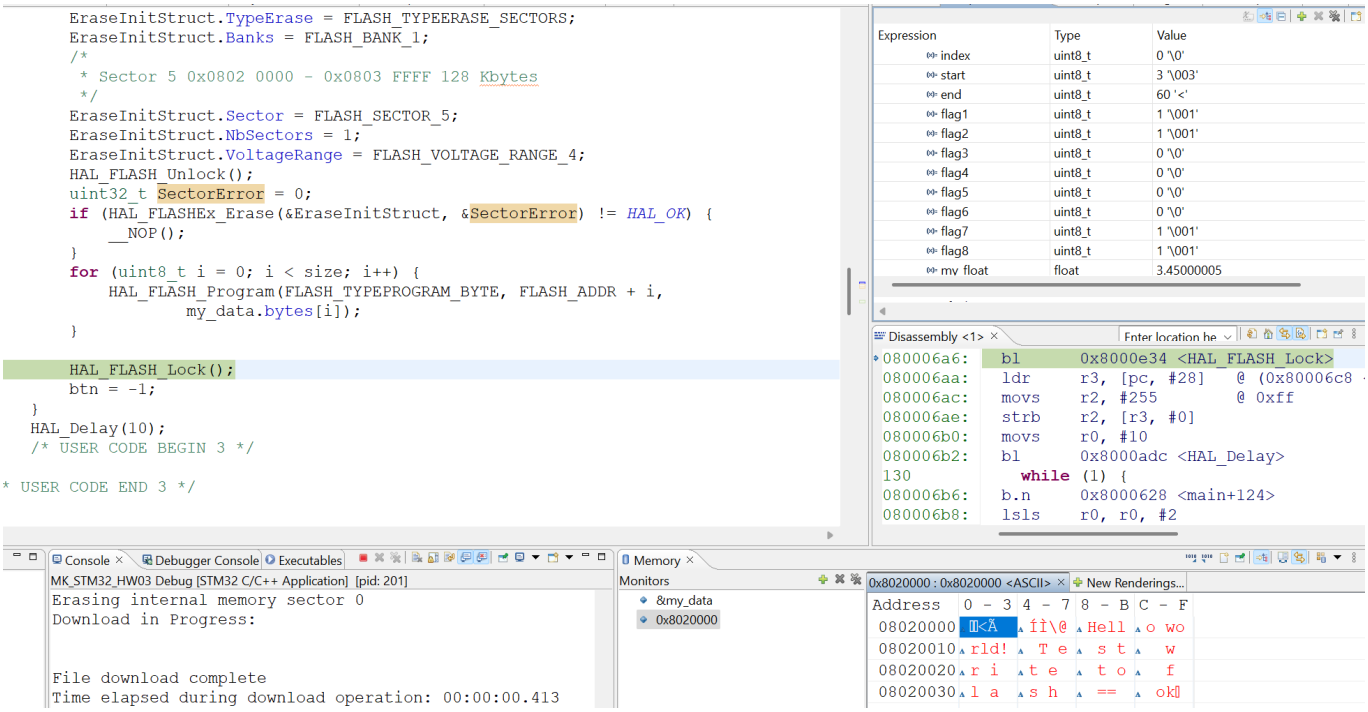


Рис. 3: Запись во flash память

Для этого напишем следующий код:

```
size = sizeof(my_data);
/* USER CODE END WHILE */
if (btn == 1) {
  static FLASH_EraseInitTypeDef EraseInitStruct;
  EraseInitStruct.TypeErase = FLASH_TYPEERASE_SECTORS;
  EraseInitStruct.Banks = FLASH_BANK_1;
  EraseInitStruct.Sector = FLASH_SECTOR_5;
  EraseInitStruct.NbSectors = 1;
  EraseInitStruct.VoltageRange = FLASH_VOLTAGE_RANGE_4;
  HAL_FLASH_Unlock();
  uint32_t SectorError = 0;
  if (HAL_FLASHEx_Erase(&EraseInitStruct, &SectorError) != HAL_OK) {
    __NOP();
  }
  for (uint8_t i = 0; i < size; i++) {
    HAL_FLASH_Program(FLASH_TYPEPROGRAM_BYTE, FLASH_ADDR + i,
        my_data.bytes[i]);
  }

  HAL_FLASH_Lock();
  btn = -1;
}
```

3. Интересные наблюдения

Во время экспериментов по записи во flash память, не получилось осуществить запись двойным словом во flash микроконтроллера. После поиска информации по этому вопросу, выяснилось, что для записи во флеш двойного слова в параллелизме, требуется подавать Vpp = 7-9в на ножку boot 0 (рисунки 4, 5).

Table 7. Program/erase parallelism

	Voltage range 2.7 - 3.6 V with External V_{PP}	Voltage range 2.7 - 3.6 V	Voltage range 2.4 - 2.7 V	Voltage range 2.1 - 2.4 V	Voltage range 1.7 V - 2.1 V
Parallelism size	x64	x32	x16		x8
PSIZE(1:0)	11	10	01		00

Note: Any program or erase operation started with inconsistent program parallelism/voltage range settings may lead to unpredicted results. Even if a subsequent read operation indicates that the logical value was effectively written to the memory, this value may not be retained.

To use V_{PP} an external high-voltage supply (between 8 and 9 V) must be applied to the V_{PP} pad. The external supply must be able to sustain this voltage range even if the DC consumption exceeds 10 mA. It is advised to limit the use of VPP to initial programming on the factory line. The V_{PP} supply must not be applied for more than an hour, otherwise the Flash memory might be damaged.

44	A5	60	94	A4	BOOT0	I	B	-	-	V_{PP}

Рис. 4: Требования для записи DoubleWord

Table 46. Flash memory programming with V_{PP} voltage (continued)

Symbol	Parameter	Conditions	Min(1)	Typ	Max(1)	Unit
V_{PP}	V_{PP} voltage range	-	7	-	9	V
I_{PP}	Minimum current sunk on the V_{PP} pin	-	10	-	-	mA
t_{VPP}(3)	Cumulative time during which V_{PP} is applied	-	-	-	1	hour

1. Guaranteed by design.
2. The maximum programming time is measured after 100K erase operations.
3. V_{PP} should only be connected during programming/erasing.

Table 11. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}–V_{SS}	External main supply voltage (including V_{DDA}, V_{DD} and V_{BAT})(1)	–0.3	4.0	V
V_{IN}	Input voltage on FT pins(2)	V_{SS}–0.3	V_{DD}+4.0	
	Input voltage on any other pin	V_{SS}–0.3	4.0	
	Input voltage for BOOT0	V_{SS}	9.0	
$\lvert\Delta V_{DDx}\rvert$	Variations between different V_{DD} power pins	-	50	mV
$\lvert V_{SSX} - V_{SS}\rvert$	Variations between all the different ground pins including V_{REF-}	-	50	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 6.3.14*		-

1. All main power (V_{DD}, V_{DDA}) and ground (V_{SS}, V_{SSA}) pins must always be connected to the external power supply, in the permitted range.
2. V_{IN} maximum value must always be respected. Refer to *Table 12* for the values of the maximum allowed injected current.

Рис. 5: Требования для записи DoubleWord

4. Eeprom emulation

Теперь проверим программный стек по эмуляции eeprom. Для этого подключим файлы eeprom.h и eeprom.c из firmware package. Работа с этой библиотекой достаточно простая. Инициализация участка памяти осуществляется функцией EE_Init(), а запись и чтение переменных производится с помощью функций EE_WriteVariable() и EE_ReadVariable().

Суть принципа сводится к тому, что на страницу памяти последовательно пишутся переменные и их виртуальные адреса. Как только страница заполнена, на следующую страницу переносятся актуальные значения переменных, а предыдущая страница затирается и так по кругу. Этот принцип обеспечивает медленный износ flash памяти (рисунок 6).

Для проверки работы напишем следующий код:

Figure 5. Page swap scheme with four pages (wear leveling)

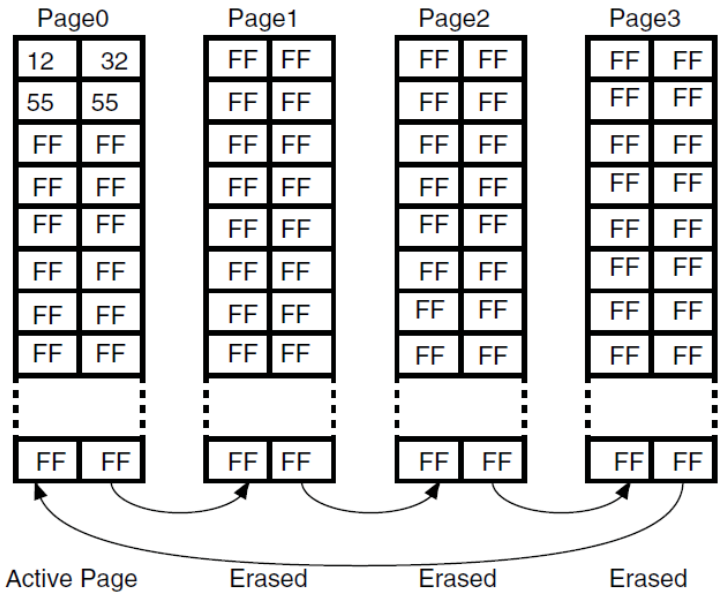
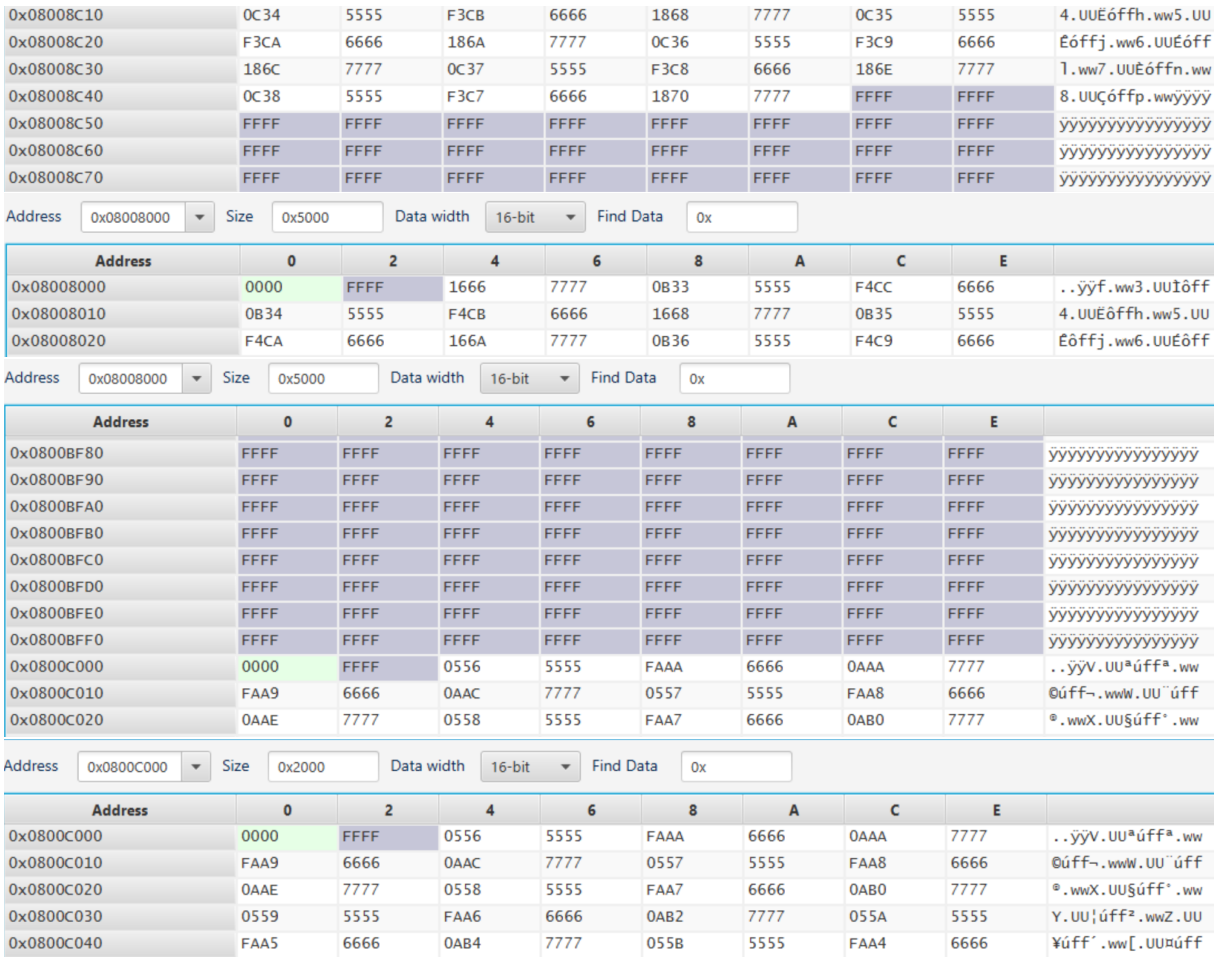


Рис. 6: Принцип ротации страниц памяти

```
/* Store 0x3000 values of Variable3 in EEPROM */
   for (VarValue = 1; VarValue <= 0x3000; VarValue++) {
    if (EE_WriteVariable(VirtAddVarTab[2], VarValue) != HAL_OK) {
     Error_Handler();
    }
    if (EE_ReadVariable(VirtAddVarTab[2], &VarDataTab[2])
      != HAL_OK) {
     Error_Handler();
    }
    if (VarValue != VarDataTab[2]) {
     Error_Handler();
    }
   }

   /* read the last stored variables data*/
   if (EE_ReadVariable(VirtAddVarTab[0], &VarDataTmp) != HAL_OK) {
    Error_Handler();
   }
   if (VarDataTmp != VarDataTab[0]) {
    Error_Handler();
   }

   if (EE_ReadVariable(VirtAddVarTab[1], &VarDataTmp) != HAL_OK) {
    Error_Handler();
   }
```

```
if (VarDataTmp != VarDataTab[1]) {
 Error_Handler();
}

if (EE_ReadVariable(VirtAddVarTab[2], &VarDataTmp) != HAL_OK) {
 Error_Handler();
}
if (VarDataTmp != VarDataTab[2]) {
 Error_Handler();
}
```

На рисунке 7 показано состояние flash памяти в разные моменты времени. Сначала, когда данные находятся на странице 1, потом когда на странице 2



0x08008C10	0C34	5555	F3CB	6666	1868	7777	0C35	5555	4.UUËóffh.ww5.UU
0x08008C20	F3CA	6666	186A	7777	0C36	5555	F3C9	6666	Êóffj.ww6.UUÉóff
0x08008C30	186C	7777	0C37	5555	F3C8	6666	186E	7777	l.ww7.UUÈóffn.ww
0x08008C40	0C38	5555	F3C7	6666	1870	7777	FFFF	FFFF	8.UUÇóffp.wwÿÿÿÿ
0x08008C50	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08008C60	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x08008C70	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

Address 0x08008000 Size 0x5000 Data width 16-bit Find Data 0x

Address	0	2	4	6	8	A	C	E	
0x08008000	0000	FFFF	1666	7777	0B33	5555	F4CC	6666	..ÿÿf.ww3.UUÌôff
0x08008010	0B34	5555	F4CB	6666	1668	7777	0B35	5555	4.UUËôffh.ww5.UU
0x08008020	F4CA	6666	166A	7777	0B36	5555	F4C9	6666	Êôffj.ww6.UUÉôff

Address 0x08008000 Size 0x5000 Data width 16-bit Find Data 0x

Address	0	2	4	6	8	A	C	E	
0x0800BF80	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BF90	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFA0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFB0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFC0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFD0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFE0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800BFF0	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0x0800C000	0000	FFFF	0556	5555	FAAA	6666	0AAA	7777	..ÿÿV.UUªúffª.ww
0x0800C010	FAA9	6666	0AAC	7777	0557	5555	FAA8	6666	©úff¬.wwW.UU¨úff
0x0800C020	0AAE	7777	0558	5555	FAA7	6666	0AB0	7777	®.wwX.UU§úff°.ww

Address 0x0800C000 Size 0x2000 Data width 16-bit Find Data 0x

Address	0	2	4	6	8	A	C	E	
0x0800C000	0000	FFFF	0556	5555	FAAA	6666	0AAA	7777	..ÿÿV.UUªúffª.ww
0x0800C010	FAA9	6666	0AAC	7777	0557	5555	FAA8	6666	©úff¬.wwW.UU¨úff
0x0800C020	0AAE	7777	0558	5555	FAA7	6666	0AB0	7777	®.wwX.UU§úff°.ww
0x0800C030	0559	5555	FAA6	6666	0AB2	7777	055A	5555	Y.UU¦úff².wwZ.UU
0x0800C040	FAA5	6666	0AB4	7777	055B	5555	FAA4	6666	¥úff´.ww[.UU¤úff

Рис. 7: Данные записаны на страницу 1, потом на страницу 2

5. Дополнительные материалы

Демонстрация работы и материалы к отчету расположены в папке на google диск по следующей ссылке: Материалы к ДЗ №03