ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНОЙ ФЛЭШ-ПАМЯТИ S29GL256 СОВМЕСТНО С МИКРОСХЕМОЙ 1892ВМ10Я

1. ВВЕДЕНИЕ

Документ описывает порядок работы с параллельной флэш-памятью S29GL256, подключенной к порту памяти (MPORT) микросхемы 1892ВМ10Я на отладочном модуле NVCom-02TEM-3U. Устройство флэш-памяти S29GL256 также применимо с микросхемами: 1892ВМ7Я, 1892ВМ8Я, 1892ВМ12АТ, 1892ВМ15АФ.

Проект MCStudio 3M для данного примера доступен на официальном сайте multicore.ru в разделе «Техподдержка → Программное обеспечение → Примеры программирования».

2. ПОРТ ВНЕШНЕЙ ПАМЯТИ НА МИКРОСХЕМЕ 1892ВМ10Я

Порт внешней памяти (MPORT) позволяет организовать обмен данными с широким набором устройств памяти и периферии. Внешний интерфейс порта обеспечивает подключение без дополнительной логики следующие типы памятей: SRAM/ROM/ EPROM/FLASH/ SDRAM/Mobile SDRAM.

Порт внешней памяти имеет следующие основные характеристики:

- • шина данных внешней памяти – 32 разряда;
- • шина адреса внешней памяти – 32 разряда;
- • программное конфигурирование типа блока памяти и его объема;
- • управление числом тактов ожидания при обмене с асинхронной памятью;
- • формирование сигналов выборки 5 блоков внешней памяти.

Более подробно порт внешней памяти (MPORT) описан в руководстве пользователя на микросхему 1892ВМ10Я.

3. ПОДКЛЮЧЕНИЕ ФЛЭШ-ПАМЯТИ S29GL256 В СОСТАВЕ МОДУЛЯ NVCOM-02TEM-3U

Микросхема флэш-памяти S29GL256 в составе модуля NVCom-02TEM-3U подключается к порту внешней памяти (MPORT), который работает в 32-разрядном режиме.

Две 16-разрядных микросхемы S29GL256 общим объемом 64 Мбайт подключены к выводу nCS[3] микросхемы 1892ВМ10Я.

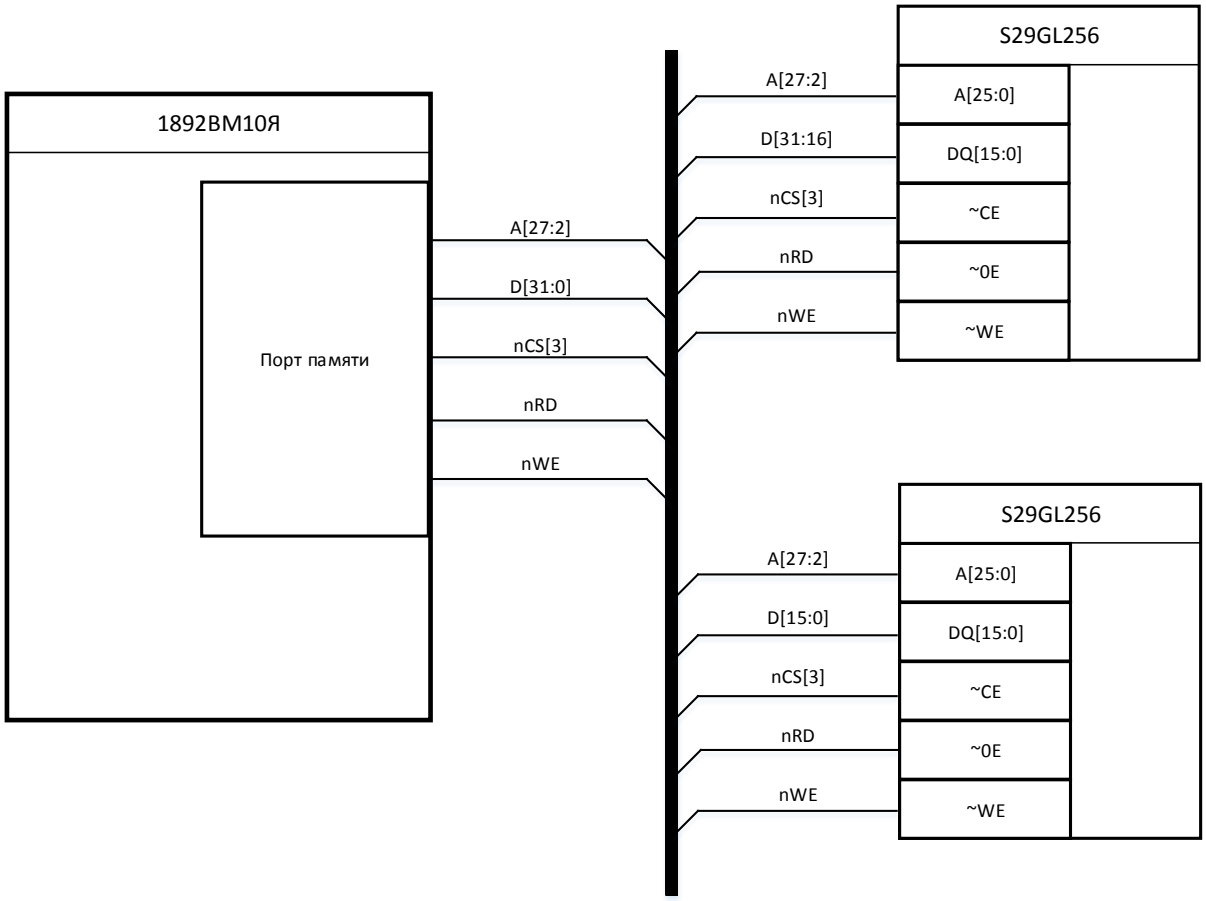

Рисунок 3.1 Структурная схема подключения 32-разрядной флэш-памяти

Младшие 16 разрядов шины данных подключены к одной 16-разрядной флэш, а старшие 16 разрядов к другой. Шина адреса, начиная со 2-го разряда, подключена параллельно к двум устройствам.

4. ЧТЕНИЕ ДАННЫХ

По умолчанию устройство находится в режиме чтения после включения питания или аппаратного сброса. Данные считываются из одной ячейки памяти за раз. Считывать содержимое ячейки можно без предварительных команд.

```
//Чтение слова
unsigned int read_single_word(unsigned int offset){
    return  *((unsigned int *)(BASE_ADDR + offset));
}
```

```
//Адресация в CPU побайтовая, а флэш адресует 4 байта, поэтому такое
соответствие 4 адреса CPU – 1 флэш

#define WORD_OFFSET 0x04

//Заметим, что (unsigned int *)(BASE_ADDR + offset) - это прибавление
численного значения стартовому адресу и последующее приведение типа к
указателю
// А (unsigned int *)BASE_ADDR + offset – это приведение значение стартового
адреса к указателю и последующее прибавление к нему численного значения
Пример: BASE_ADDR  = 0xBFC00000
(unsigned int *)(BASE_ADDR + 0x1) указывает на BFC00001 (приведет к ошибке)
(unsigned int *)BASE_ADDR  + 0x1  указывает на BFC00004
```

5. ЧТЕНИЕ ID

Режим Autoselect предоставляет идентификатор производителя, идентификацию устройства и информацию защите сектора, через коды идентификаторов, выводимые из внутреннего регистра (отдельно от массива памяти) на DQ7-DQ0.

```
//Включение режима Autoselect
void S29GL256_autoselect_mode_en(){
    if(!Global_Bypass_mode) S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_AS;

}
```

Для того чтобы прочитать ID устройства необходимо включить режим Autoselect и прочитать коды идентификаторов.

```
//Чтение ID устройства
uint S29GL256_read_device_id(){
    S29GL256_autoselect_mode_en();
    uint id = S29GL256_read_single_word(BASE_ADDR + 0xE*WORD_OFFSET);
    S29GL256_reset_flash();
    return id;
}
```

Коды идентификатора ID флэш можно получить командой чтения слова по адресу BASE_ADDR + 0xE*WORD_OFFSET. Где:

```
#define BASE_ADDR 0xBFC00000
#define WORD_OFFSET 0x04
```

6. СТАТУСНЫЙ РЕГИСТР

По умолчанию устройство флэш находится в режиме чтения. При использовании команды чтения на выходе будет получено значение, содержащееся в ячейке памяти. Однако, некоторые операции, такие как запись данных и очистка памяти, возвращают управление флэш-памятью не сразу. В таком случае при попытке прочитать данные на выходе будет получено значение статусного регистра флэш-памяти. Опрос статусного регистра означает чтение и проверку битов статусного регистра на наличие признака завершения операции или ошибки.

В документации S29GL256 приведен алгоритм опроса, ниже представлена возможная реализация.

```
//Опрос статусных регистров чипа
OpStatus S29GL256_chip_polling(uint _isWBP){

    uint Read_1 = S29GL256_read_single_word(BASE_ADDR);
    uint Read_2 = S29GL256_read_single_word(BASE_ADDR);
    uint Read_3 = S29GL256_read_single_word(BASE_ADDR);
    if( (DQ(6,Read_1) != DQ(6,Read_2))  &&
        (DQ(6,Read_2) != DQ(6,Read_3))  ){//DQ6 Toggles between Read_1
& Read_2 and Read_2 & Read_3 "Yes"
        if( (DQ(1,Read_1) == DMASK(1)) && (_isWBP == 1)
){//WriteBuffer program and Read_1 DQ1 is set "Yes"
            return WRITE_ABORT;
        }else{//Read_1 DQ1 is set "No"
            if( DQ(5,Read_1) == DMASK(5)){//Read_1 DQ5 is set "Yes"
ret timeout
                return TIMEOUT;
            }else{//Read_1 DQ1 is set "No" ret to start
                return BUSY;
            }
        }
    }else{//DQ6 Toggles between Read_1 & Read_2 and Read_2 & Read_3
"No" new read
        Read_1 = S29GL256_read_single_word(BASE_ADDR);
        Read_2 = S29GL256_read_single_word(BASE_ADDR);
        if((DQ(2,Read_1) != DQ(2,Read_2))){//DQ2 Toggles "Yes" ret
suspend
            return SUSPEND;
        }else{//DQ2 Toggles "No" ret done
            return DONE;
        }
    }
}
```

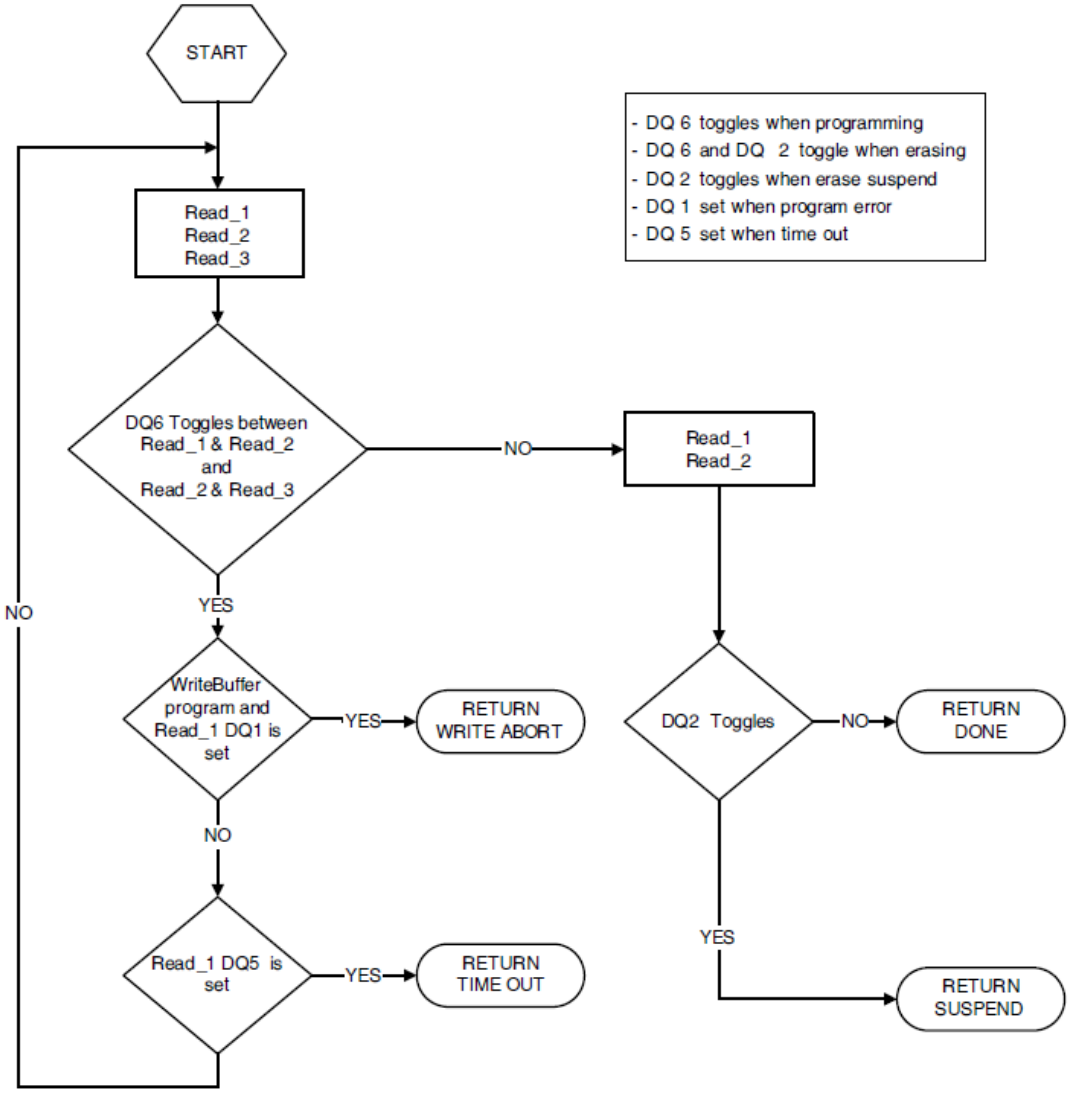

Рисунок 6.1 Общий алгоритм опроса статусного регистра

7. РАЗБЛОКИРОВКА УСТРОЙСТВА

Особенность организации микросхем флэш-памяти заключается в том, что в них нельзя произвести запись тем же способом, что и в обычное статическое ОЗУ. Для записи или стирания данных во флэш необходимо произвести разблокировку. Для этого необходимо подать на флэш команды.

В случае с S29GL256 в 16-разрядном режиме:

- первый цикл разблокировки – это слово 0x00AA, записанное по адресу 0x555;
- второй цикл – это слово 0x0055, записанное по адресу 0x2AA.

```
//Стартовый адрес
#define BASE_ADDR 0xBFC00000
#define OFFSET_CMD_0  0x000
#define OFFSET_CMD_1  0x555

#define CMD_WUC1          0x00AA00AA //Команда снятия блокировки 1
#define CMD_WUC2          0x00550055 //Команда снятия блокировки 2

//Разблокировка флэш
void S29GL256_unlock_flash(){
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_WUC1;
    *( (uint *)BASE_ADDR + OFFSET_CMD_2 ) = CMD_WUC2;
}
```

К шине данных подключено две 16-разрядные микросхемы флэш-памяти. Команды снятие блокировки дублируются. Младшие 2 байт - команда для одной микросхемы, старшие 2 байта - для другой (0x00AA_00AA).

7.1 Режим BYPASS

Режим BYPASS предназначен для более быстрого программирования флэш-памяти. При включенном режиме BYPASS отпадает необходимость каждый раз вводить первую команду снятия блокировки.

```
#define OFFSET_CMD_1  0x555
#define CMD_BP_EN          0x00200020 //Команда входа в режим BYPASS
#define CMD_BP_RST1        0x00900090 //Команды сброса режима BYPASS

//Индикатор режима BYPASS
static uint Global_Bypass_mode = 0;

//Включение режима BYPASS
OpStatus S29GL256_unlock_bypass(){
    if(Global_Bypass_mode) return FAIL;
    S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_BP_EN;
    Global_Bypass_mode  = 1;
    return PASS;
}
```

8. ЗАПИСЬ ДАННЫХ

8.1 Запись одного слова

Порядок работы:

- снятие блокировки (если не включён режим BYPASS);
- подача команды записи одного слова (CMD_WRSW 0x00A000A0);
- запись данных в нужный адрес (запись в режиме одного слова не ограничивается адресами одного сектора);
- ожидание завершения.

```
//Запись одного слова
OpStatus S29GL256_write_single_word(uint _addr, uint _word){
    if(!Global_Bypass_mode) S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_WRSW;
    *( (uint *)(_addr) ) = _word;

    //Опрос статуса чипа, если операция завершена - выход из функции
    while(1){
        OpStatus st = S29GL256_chip_polling(0);
        if(st == BUSY) continue;
        else if(st == DONE) return PASS;
        else return FAIL;
    }
}
```

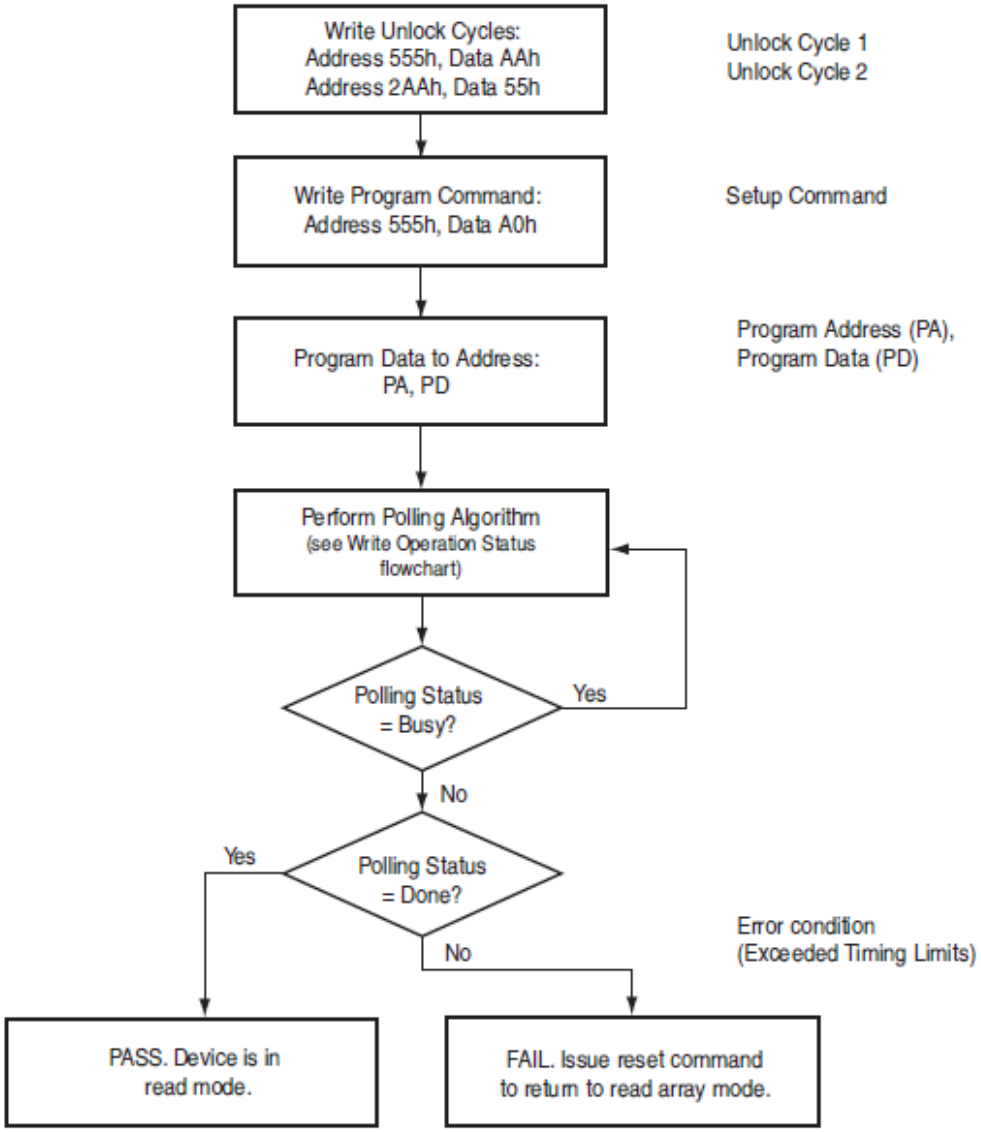

Рисунок 8.1 Алгоритм опроса статусного регистра при записи одного слова

8.2 Запись буфера

Порядок работы:

- снятие блокировки (если не включён режим BYPASS);
- подача команды записи буфера (CMD_WRBUF_LOAD 0x00250025);
- запись значения с количеством слов;
- поочередная запись каждого слова;
- запись команды подтверждения (CMD_WRBUF_CONFIRM 0x00290029);
- ожидание завершения.

```
//Запись буфера
OpStatus S29GL256_write_buffer(uint _sector_addr, uint* _buf, uint
_word_cnt){
    if(_word_cnt > MAX_BUF_LEN) return WRITE_ABORT;
    if(!Global_Bypass_mode) S29GL256_unlock_flash();

    *( (uint *)_sector_addr ) = CMD_WRBUF_LOAD;
    *( (uint *)_sector_addr ) = ((_word_cnt - 1)<<16) | (_word_cnt -
1);

    //Поочередная запись значений
    uint i = 0;
    for(i = 0; i<_word_cnt; i++)
        *((uint *)(_sector_addr+ i*WORD_OFFSET)) =  *(_buf + i  );

    *( (uint *)_sector_addr ) = CMD_WRBUF_CONFIRM;

    //Опрос статуса чипа, если операция завершена - выход из функции
    while(1){
        if(S29GL256_chip_polling(1) == DONE){
            return PASS;
        }else continue;
    }
}
```

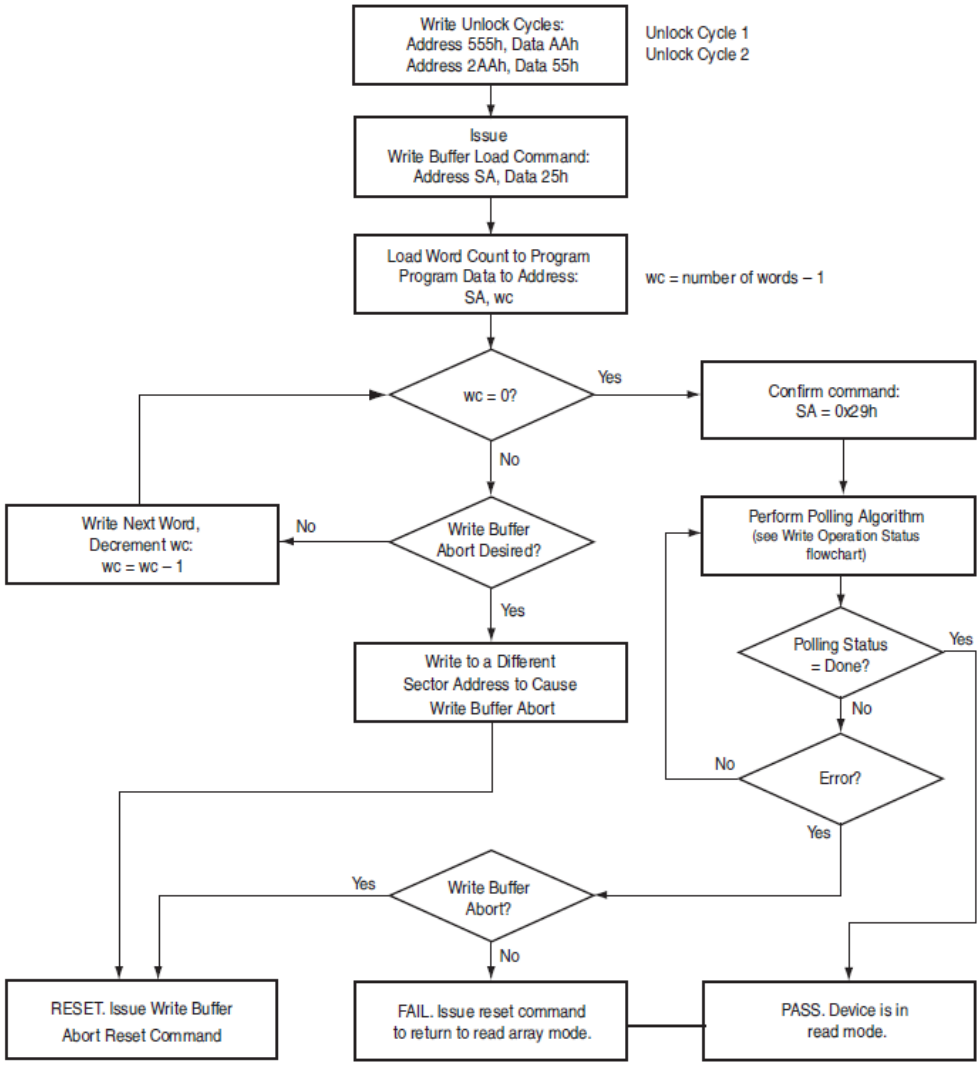

Рисунок 8.2 Алгоритм опроса статусного регистра при записи буфера

Программа записи через буфер предусматривает возможность приостановки работы.

```
#define CMD_Suspend        0x00B000B0 //Команда паузы

//Приостановка выполнения операции
void suspend(){
    *( (unsigned int *)BASE_ADDR ) = CMD_Suspend; // write suspend
command
}
```

Во время паузы можно считывать данные любого другого сектора, кроме того, в который записывался буфер. Доступна команда Autoselect.

Для продолжения приостановленной программы нужно подать команду возобновления работы.

```
#define CMD_Resume         0x00300030 //Команда продолжить

//Возобновление выполнения заданных действий
void resume_program(unsigned int sector_addr){
    *( (unsigned int *)BASE_ADDR  ) = CMD_Resume; // write resume
command
}
```

9. ОЧИСТКА ПАМЯТИ

9.1 Очистка сектора

Порядок работы:

- снятие блокировки (если не включён режим BYPASS);;
- подача команды доступа (CMD_SETUP 0x00800080);
- повторное снятие блокировки;
- поочередная подача команды очистки сектора (CMD_ERSEC 0x00300030) по адресам начала секторов, которые нужно очистить.

В последнем пункте нет ограничения по количеству стираемых секторов, однако команда в адрес каждого последующего сектора должна быть подана не позже, чем через 50 µсек после подачи предыдущей. Иначе инструкция начнет выполняться и последующие команды (кроме паузы) будут игнорироваться.

```
//Очистка сектора
OpStatus S29GL256_erase_sector(uint *_sector_addr_array, uint _len){
    if(!Global_Bypass_mode) S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_SETUP; //Setup Command
    S29GL256_unlock_flash();
    int i;
    for(i = 0; i < _len; i++)
        *( (uint *)_sector_addr_array[i] ) = CMD_ERSEC; // Erase
sector

    //Опрос статуса чипа, если операция завершена - выход из функции
    while(1){
        if(  DQ(3, *((uint *)_sector_addr_array[0])) == DMASK(3)){
//Poll DQ3. DQ3 = 1?
            while(1){
                if(S29GL256_chip_polling(0) == DONE){
                    return PASS;
                }else{
                    if(DQ(5, *((uint *)_sector_addr_array[0])) ==
DMASK(5))//DQ5 = 1?
                        return FAIL;
                    else continue;
                }
            }
        }else continue;
    }
}
```

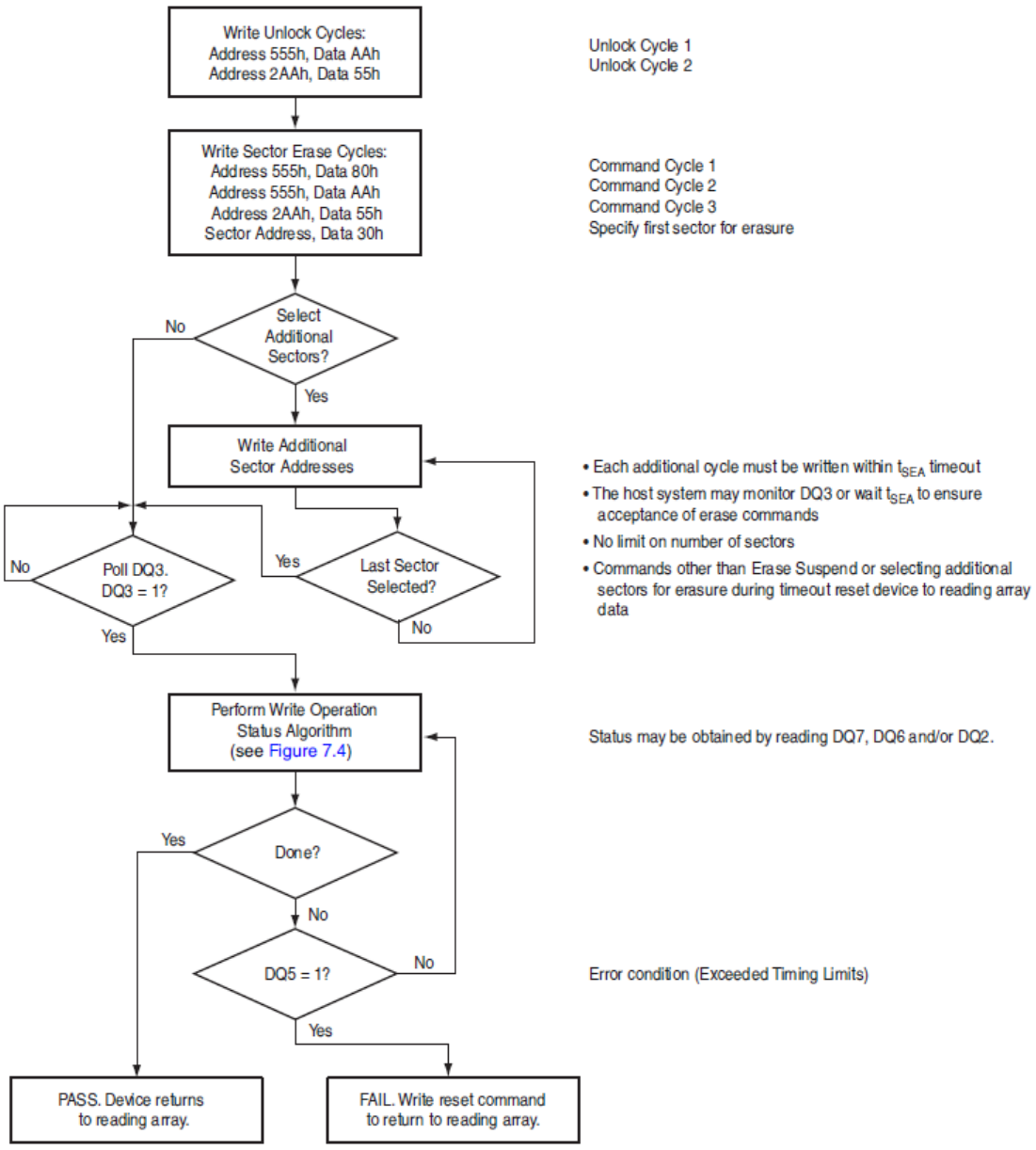

Рисунок 9.1 Алгоритм опроса статусного регистра при очистке сектора

Существует возможность приостановить программу очистки сектора. Команда полностью аналогична той, что приостанавливает работу записи буфера.

```
#define CMD_Suspend        0x00B000B0 //Команда паузы

//Приостановка выполнения операции
void suspend(){
    *( (unsigned int *)BASE_ADDR ) = CMD_Suspend; // write suspend
command
}
```

Во время паузы можно считывать данные любого другого сектора, кроме того, который должен быть очищен. Доступна команда Autoselect.

Для продолжения приостановленной программы нужно подать команду возобновления работы.

```
#define CMD_Resume         0x00300030 //Команда продолжить


//Возобновление очистки данных
void resume_erase(unsigned int sector_addr){
    *( (unsigned int *)sector_addr ) = CMD_Resume; // write resume
command
}
```

```
Различие от возобновления записи буфера в том, что команда CMD_Resume
подается в адрес сектора(sector_addr), на котором было приостановлена
очистка.
```

9.2 Очистка флэш

Порядок работы:

- снятие блокировки (если не включён режим BYPASS);;
- подача команды доступа (CMD_SETUP 0x00800080);
- повторное снятие блокировки;
- запись команды очистки чипа (CMD_ERCHIP 0x00100010) в базовый адрес флэш (BASE_ADDR)

```
//Очистка всего чипа
OpStatus S29GL256_erase_chip(){
    if(!Global_Bypass_mode) S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1 ) = CMD_SETUP;
    S29GL256_unlock_flash();
    *( (uint *)BASE_ADDR + OFFSET_CMD_1  ) = CMD_ERCHIP;

    //Опрос статуса чипа, если операция завершена - выход из функции
    while(1){
        if(S29GL256_chip_polling(0) == DONE)
            return PASS;
        else{
            if(DQ(5, *((uint *)BASE_ADDR)) == DMASK(5))//DQ5 = 1?
                return FAIL;
            else continue;
        }
    }

}
```

Алгоритм опроса аналогичен приведённому при рассмотрении очистки сектора.

10. ПРИМЕР РАБОТЫ

Чтение ID устройства:

```
int main(void){

    //Чтение id устройства
    //id = 0x22222222 - GL256P
    uint id = S29GL256_read_device_id();
```

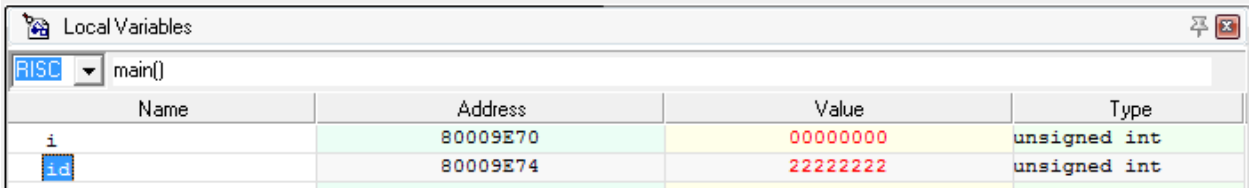

Рисунок 10.1 Результат, полученный функцией чтения ID

Запись методом одиночного слова и последующая проверка:

```
    //Объявление адреса сектора
    //Порядковый номер сектора
    uint j = 0;
    //Адрес сектора
    uint sector_addr = FIRST_SECTOR_ADDR + j*SECTOR_OFFSET;

    //Очистка сектора
    S29GL256_erase_sector(&sector_addr, 1);

    //Включение режима bypass
    S29GL256_unlock_bypass();

    //Заполнение буферов
    uint i;
    uint td_32b[D_32b_LEN];
    uint rd_32b[D_32b_LEN];
    array32_fill(td_32b, D_32b_LEN) ;

    //Запись методом одиночного слова
    for(i = 0; i<D_32b_LEN; i++)
        S29GL256_write_single_word(sector_addr + i*WORD_OFFSET,
td_32b[i]);

    //Чтение записанных данных и верификация
    for(i = 0; i<D_32b_LEN; i++)
        rd_32b[i] = S29GL256_read_single_word(sector_addr +
i*WORD_OFFSET);

    uint ok = array32_compare(td_32b, rd_32b,D_32b_LEN) ;
```

Local Variables

RISC main()

Name	Address	Value	Type
i	80009EF0	16	unsigned int
id	80009EF4	572662306	unsigned int
j	80009EF8	0	unsigned int
ok	80009EFC	1	unsigned int
⊟ rd_32b	80009F44	[ ]	unsigned int[16]
[0]	80009F44	1	unsigned int
[1]	80009F48	2	unsigned int
[2]	80009F4C	3	unsigned int
[3]	80009F50	4	unsigned int
[4]	80009F54	5	unsigned int
[5]	80009F58	6	unsigned int
[6]	80009F5C	7	unsigned int
[7]	80009F60	8	unsigned int
[8]	80009F64	9	unsigned int
[9]	80009F68	10	unsigned int
[10]	80009F6C	11	unsigned int
[11]	80009F70	12	unsigned int
[12]	80009F74	13	unsigned int
[13]	80009F78	14	unsigned int
[14]	80009F7C	15	unsigned int
[15]	80009F80	16	unsigned int
sector_addr	80009F00	3154116608	unsigned int
⊟ td_32b	80009F04	[ ]	unsigned int[16]
[0]	80009F04	1	unsigned int
[1]	80009F08	2	unsigned int
[2]	80009F0C	3	unsigned int
[3]	80009F10	4	unsigned int
[4]	80009F14	5	unsigned int
[5]	80009F18	6	unsigned int
[6]	80009F1C	7	unsigned int
[7]	80009F20	8	unsigned int
[8]	80009F24	9	unsigned int
[9]	80009F28	10	unsigned int
[10]	80009F2C	11	unsigned int
[11]	80009F30	12	unsigned int
[12]	80009F34	13	unsigned int
[13]	80009F38	14	unsigned int
[14]	80009F3C	15	unsigned int
[15]	80009F40	16	unsigned int

Рисунок 10.2 Результат записи методом одиночного слова

Проверка метода записи через буфер:

```
    //Очистка сектора
    S29GL256_erase_sector(&sector_addr, 1);

    //Метод записи через буфер
    S29GL256_write_buffer(sector_addr, td_32b, D_32b_LEN);

    //Чтение записанных данных и верификация
    for(i = 0; i<D_32b_LEN; i++)
        rd_32b[i] = S29GL256_read_single_word(sector_addr +
i*WORD_OFFSET);
```

```
    ok = array32_compare(td_32b, rd_32b, D_32b_LEN) ;

    //Очистка чипа
    S29GL256_erase_chip();

    //Выключение режима BYPASS
    S29GL256_bypass_exit();
```

Local Variables

RISC main()

Name	Address	Value	Type
i	80009EF0	00000010	unsigned int
id	80009EF4	22222222	unsigned int
j	80009EF8	00000000	unsigned int
ok	80009EFC	00000001	unsigned int
⊟ rd_32b	80009F44	[ ]	unsigned int[16]
[0]	80009F44	00000001	unsigned int
[1]	80009F48	00000002	unsigned int
[2]	80009F4C	00000003	unsigned int
[3]	80009F50	00000004	unsigned int
[4]	80009F54	00000005	unsigned int
[5]	80009F58	00000006	unsigned int
[6]	80009F5C	00000007	unsigned int
[7]	80009F60	00000008	unsigned int
[8]	80009F64	00000009	unsigned int
[9]	80009F68	0000000A	unsigned int
[10]	80009F6C	0000000B	unsigned int
[11]	80009F70	0000000C	unsigned int
[12]	80009F74	0000000D	unsigned int
[13]	80009F78	0000000E	unsigned int
[14]	80009F7C	0000000F	unsigned int
[15]	80009F80	00000010	unsigned int
sector_addr	80009F00	BC000000	unsigned int
⊟ td_32b	80009F04	[ ]	unsigned int[16]
[0]	80009F04	00000001	unsigned int
[1]	80009F08	00000002	unsigned int
[2]	80009F0C	00000003	unsigned int
[3]	80009F10	00000004	unsigned int
[4]	80009F14	00000005	unsigned int
[5]	80009F18	00000006	unsigned int
[6]	80009F1C	00000007	unsigned int
[7]	80009F20	00000008	unsigned int
[8]	80009F24	00000009	unsigned int
[9]	80009F28	0000000A	unsigned int
[10]	80009F2C	0000000B	unsigned int
[11]	80009F30	0000000C	unsigned int
[12]	80009F34	0000000D	unsigned int
[13]	80009F38	0000000E	unsigned int
[14]	80009F3C	0000000F	unsigned int
[15]	80009F40	00000010	unsigned int

Рисунок 10.3 Результат проверки метода записи через буфер