Проектирование схем микроэлектронных устройств в Proteus с использованием внешней памяти.
Часть 2

Татьяна Колесникова (beluikluk@gmail.com)

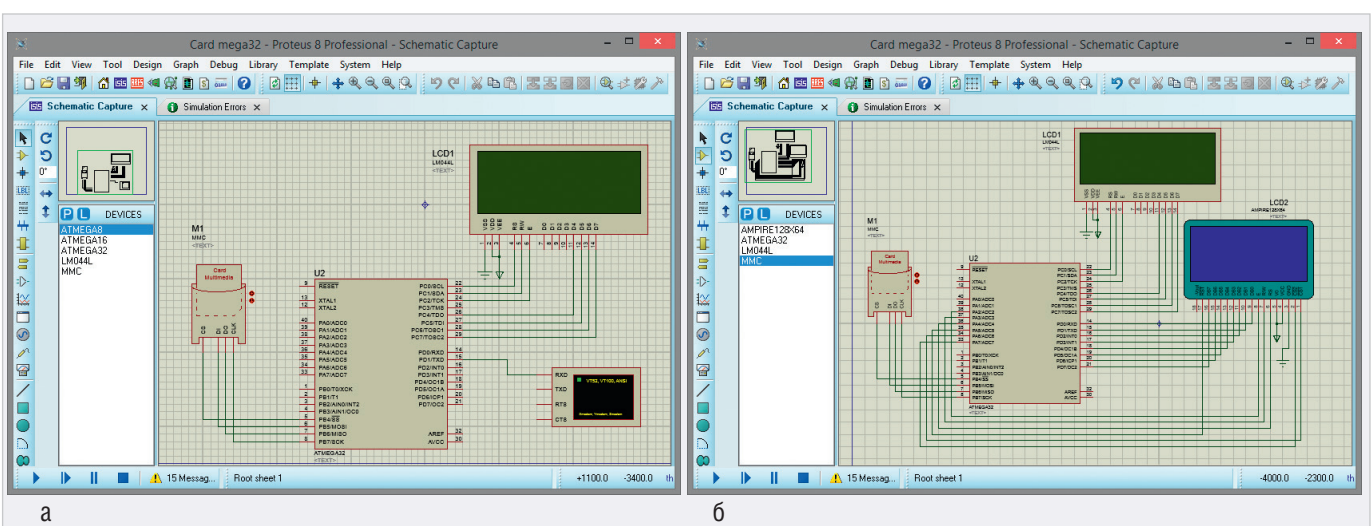
В настоящее время большинство электронных устройств проектируют с применением микроконтроллеров. Однако зачастую для реализации задуманного устройства памяти микроконтроллера может не хватить. В таких случаях выполняют её расширение, например, за счёт карты памяти. Для проверки работоспособности схемы удобно применить программу-эмулятор Proteus, чья библиотека содержит микроконтроллеры с возможностью их программирования, карту MMC, а также другие цифровые компоненты и устройства вывода информации.

В устройствах на микроконтроллерах для хранения больших объёмов информации применяют внешнюю память. Если требуется хранить единицы мегабайт, то подойдут микросхемы последовательной Flash-памяти. Для больших объёмов (десятки и сотни мегабайт) используют карты памяти. В этой статье речь пойдёт о подключении внешней памяти MMC (карты MultiMedia Card) к микроконтроллеру и её управлении программным способом. Сначала мы соберём схему на основе микроконтроллера ATmega32, подключим к нему периферийные устройства (графический и алфавитно-цифровой дисплей, виртуальный терминал) для вывода считанной с карты информации, создадим интерфейс обмена данными через шину SPI. Затем разберём функции для работы с файлами и функции графической библиотеки CodeVisionAVR. Применим их для управления файловой структурой карты (переименования, создания, удаления файлов и папок на карте), записи и чтения фиксированного числа байт или всей текстовой или графической информации из размещённого во внешней памяти файла и их вывода на дисплей или терминал в эмуляторе электронных схем Proteus.

Проектирование схемы электрической принципиальной в Proteus

Рассмотрим работу с картой памяти в Proteus на примере устройства обмена информацией, работающего под управлением микроконтроллера AVR, написание программы инициализации которого выполнено с помощью CodeVisionAVR. В спроектированном устройстве карту применяют для расширения памяти и создания интерфейса обмена информацией через шину SPI, которая присутствует во многих микроконтроллерах, в частности, и в микроконтроллерах AVR семейства Mega. Вывод считанной с карты памяти информации выполним на экран терминала, алфавитно-цифрового и графического дисплеев. Создание схемы в редакторе ISIS (см. рис. 1а) было рассмотрено в [1], поэтому подробно остановимся только на подключении к микроконтроллеру ATmega32 графического дисплея (см. рис. 1б), в качестве которого применим микросхему AMPIRE128X64 с разрешением экрана 128×64 пикселя. Микросхема работает под управлением контроллера KS0108, который принимает и обрабатывает команды управления и выводит соответствующую графику на дисплей. Контроллер KS0108 не имеет своего знакогенератора. В нашем примере его функции (вывод текста и графики), а также управление микросхемой выполнит программа микроконтроллера.

Выбор компонентов из базы данных для последующего их размеще-



а б

Рис. 1. Сопряжение в рабочей области редактора ISIS программы Proteus микроконтроллера ATmega32 с картой памяти и устройствами вывода информации: алфавитно-цифровым дисплеем LM044L и терминалом (а), алфавитно-цифровым дисплеем LM044L и графическим дисплеем AMPIRE128×64 (б)

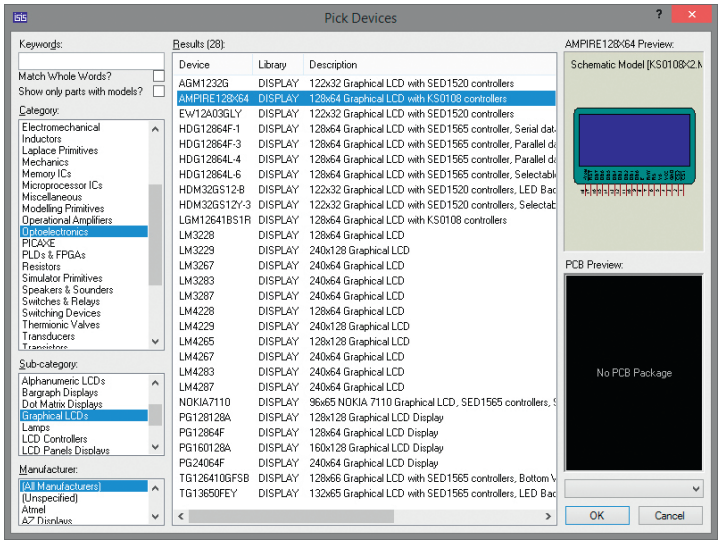


Рис. 2. Раздел Graphical LCDs библиотеки Optoelectronics

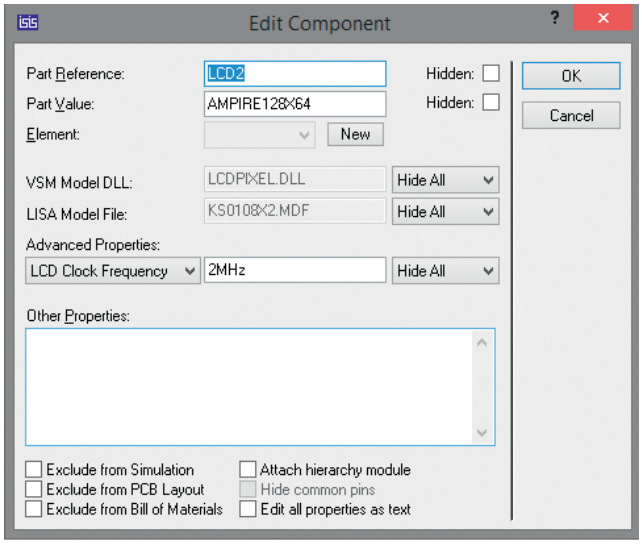


Рис. 3. Окно свойств графического дисплея AMPIRE128×64

ния в рабочей области редактора ISIS выполняют в окне Pick Devices, которое открывают командой контекстного меню Place/Component/From Libraries или нажатием кнопки P на панели DEVICES (по умолчанию панель расположена в левой части программы и содержит список имеющихся в проекте компонентов). Открывают панель DEVICES нажатием кнопки Component Mode на левой панели инструментов редактора ISIS.

Микросхема графического дисплея AMPIRE128×64 находится в разделе Graphical LCDs библиотеки Optoelectronics (см. рис. 2). Графические дисплеи обеспечивают создание на своём экране матрицы точек, высвечивающих изображения и тексты, и обладают большей гибкостью в отличие от алфавитно-цифровых модулей, жёстко фиксирующих размеры и положение символов. Графические модули не накладывают сколь-либо серьёзных ограничений на отображаемую информацию, причём это могут быть не только символы алфавита, но и специальные символы, графики, диаграммы, элементы оформления.

Использование графического дисплея расширяет область применения устройства, повышает уровень его информативности, предоставляет большие возможности при отображении данных относительно применения алфавитно-цифрового дисплея. Информация отображается на экране модуля дисплея поточечно, что позволяет получить любое необходимое изображение. На каждую точку экрана приходится один информационный бит, который управляет свечением пикселя.

Каких-либо стандартных правил сопряжения микроконтроллеров с дисплеями не существует, и в каждом конкретном случае сопряжение может выполняться по-разному.

Микросхема AMPIRE128X64 имеет 18 контактов, назначение которых следующее:

- GND – «земля»;
- V_{cc} – напряжение питания +5 В;
- V0 – напряжение контрастности от 0 до +5 В (настройка контрастности экрана);
- RS – установка режима приёма информации (RS = 1 – данные, RS = 0 – команды);
- R/W – выбор операции чтения (R/W = =1) или записи (R/W = 0);
- E – линия синхронизации;
- DB0...DB7 – шина данных/команд;
- $-V_{out}$ – выход отрицательного напряжения;
- $\overline{CS\,1}$ – активация левого сегмента дисплея ($\overline{CS\,1}$ = 0 – сегмент активный, $\overline{CS\,1}$ = = 1 – сегмент не активный);
- $\overline{CS\ 2}$ – активация правого сегмента дисплея ($\overline{CS\,2}$ = 0 – сегмент активный, $\overline{CS\,2}$ = 1 – сегмент не активный);
- $\overline{RST}$ – сигнал сброса контроллера дисплея.

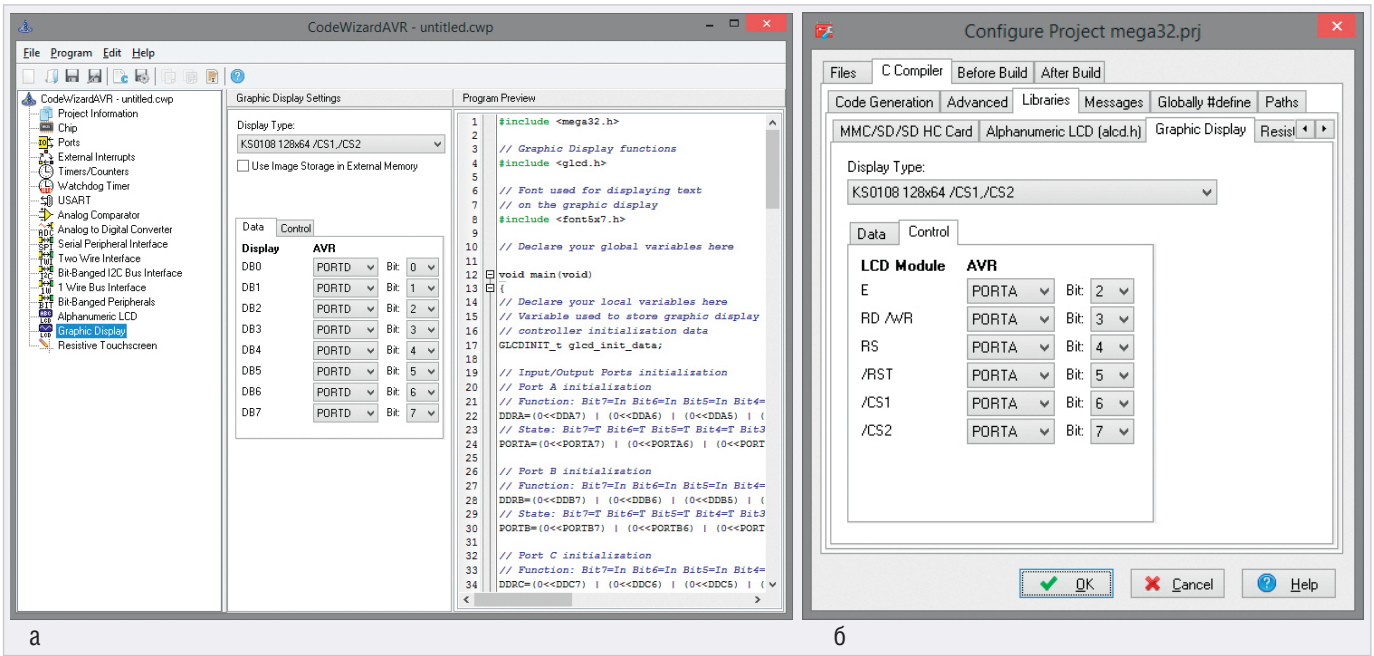
Для подключения микросхемы AMPIRE128×64 к схеме управления используются параллельная синхронная шина данных/команд (DB0...DB7), вывод выбора операции чтения/записи (R/W), вывод выбора регистра данных/команд (RS), вывод синхронизации (E), выводы активизации сегментов дисплея ($\overline{CS\,1}$ и $\overline{CS\,2}$) и вывод $\overline{RST}$, на который подаётся сигнал сброса контроллера дисплея. Подсоединим выводы модуля дисплея DB0...DB7 к выводам PD0...PD7, выводы E, R/W и RS к выводам PA2...PA4, а выводы $\overline{RST}$, $\overline{RS\ 1}$ и $\overline{RS\ 2}$ к выводам PA5...PA7 микроконтроллера ATmega32 так, как показано на рис. 1б.

Выводы PA0, PA1 микроконтроллера не рекомендуется использовать для подключения дисплея из-за возникновения внутреннего конфликта с картой памяти, о чём в CodeVisionAVR при подключении к проекту дисплея выводится предупреждающее сообщение.

Выводы GND и V_{cc} подключим к «земле» и напряжению +5 В соответственно. На вывод V0 подаётся напряжение контрастности (от 0 до +5 В). На практике этот вывод подключают к питанию через подстроечный резистор, который позволяет плавно регулировать контрастность отображения символов на дисплее. Символы «земли» и питания добавляют в схему, выбрав на панели TERMINALS строки GROUND и POWER. Панель открывают нажатием кнопки Terminals Mode на левой панели редактора ISIS.

Выбор линий портов микроконтроллера для подключения к указанным выводам дисплея выполняется произвольно. В окне свойств дисплея (окно открывают двойным щелчком левой кнопки мыши после его выделения на схеме) в поле Advanced Properties из выпадающего списка выбирают пункт LCD Clock Frequency – тактовая частота (см. рис. 3), значение которой должно совпадать с частотой работы микроконтроллера (в нашем примере – 2 МГц).

Приём информации микросхемой AMPIRE128×64 осуществляется по 8-разрядной шине данных/команд. Подача управляющих сигналов через подключённые к портам микроконтролле-



а б

Рис. 4. Настройка параметров графического дисплея: в окне CodeWizardAVR (а), на вкладке Graphic Display окна Configure Project (б)

ра ATmega32 линии выполняется программно.

После создания схемы, подключения всех приборов и настройки их параметров переходят к следующему этапу разработки: написанию программного кода управления устройством в CodeVisionAVR. В результате его компиляции (при условии отсутствия в коде ошибок) на диске компьютера будет получен hex-файл, путь к которому указывают в окне свойств микроконтроллера в Proteus.

Завершающим этапом работы в Proteus является запуск процесса моделирования схемы в редакторе ISIS, который выполняют кнопкой Run the simulation, расположенной в левом нижнем углу окна редактора или командой основного меню Debug/Run Simulation.

Создание программного кода в CodeVisionAVR

Формирование программного кода в CodeVisionAVR выполняют при помощи автоматического генератора CodeWizardAVR или вручную с нуля, используя синтаксис языка программирования С и функции стандартных библиотек программы. Удобство применения генератора состоит в быстром получении кода выполнения функций инициализации микроконтроллера и его портов ввода/вывода, аналогового компаратора, таймеров/счётчиков, интерфейса UART и SPI, алфавитно-цифровых и графических дисплеев и др. Однако в процессе работы мастера формируется достаточно объёмный код, который впоследствии приходится редактировать.

После создания командой основного меню File/New/Project нового проекта в CodeVisionAVR открывается окно генератора кода CodeWizardAVR, где задают параметры микроконтроллера, его внутренних ресурсов и используемых в схеме периферийных устройств, что уже было рассмотрено в [1], поэтому подробно остановимся на определении свойств графического дисплея. В окне генератора кода (см. рис. 4а) перейдём на вкладку Graphic Display Settings и укажем тип контроллера, разрешение дисплея и инверсию выводов активации сегментов дисплея (поле Display Type, в нашем примере – KS0108 128×64 /CS1, /CS2), применение внешней памяти для хранения изображения (Use Image Storage in External Memory, в нашем примере флажок в чекбоксе снят). На вкладках Data и Control настраивают параметры подключения микроконтроллера (порт и номер вывода) к микросхеме дисплея – в нашем примере биты 2...7 порта PA микроконтроллера подключены к выводам E, R/W, RS, $\overline{RST}$, $\overline{CS\,1}$ и $\overline{CS\,2}$ дисплея соответственно, биты 0...7 порта PD микроконтроллера подключены к выводам DB0...DB7 дисплея.

Также настройку параметров дисплея можно выполнить на базе уже существующего проекта CodeVisionAVR, для чего с помощью команды Project/Configure основного меню открывают окно Configure Project. В этом окне следует перейти на вкладку C Compiler и затем – на вкладку Libraries, где открывают вкладку Graphic Display (см. рис. 4б) и устанавливают в поле Display Type тип контроллера, разрешение дисплея и инверсию выводов активации сегментов дисплея. Подключение управляющих сигналов и сигналов данных/команд определяют на вкладках Control и Data.

Прежде чем приступить к написанию программного кода в CodeVisionAVR, подключим поддержку карты памяти MMC и увеличим размер стека, для чего на вкладке Libraries окна Configure Project откроем вкладку MMC/SD/SD HC Card и установим флажок в чекбоксе Enable MMC/SD/SD HC Card and FAT Support (разрешить поддержку карт памяти и файловой системы FAT), см. рис. 5а. На вкладке C Compiler откроем вкладку Code Generation (см. рис. 5б) и в поле Data Stack Size укажем размер стека в байтах – для компиляции кода в нашем примере значения 1920 будет достаточно.

Применение функций библиотеки ff.h для записи данных во внешнюю память, их чтения и отображения на экране терминала и алфавитно-цифрового дисплея

Для работы с картами памяти MMC/SD/SD HC, отформатированными в FAT12, FAT16 или FAT32, в CodeVisionAVR предусмотрены библиотеки функций sdcard.h и ff.h, описание которых было рассмотрено в [1]. Продемонстрируем на конкретном примере работу с функциями создания фай-

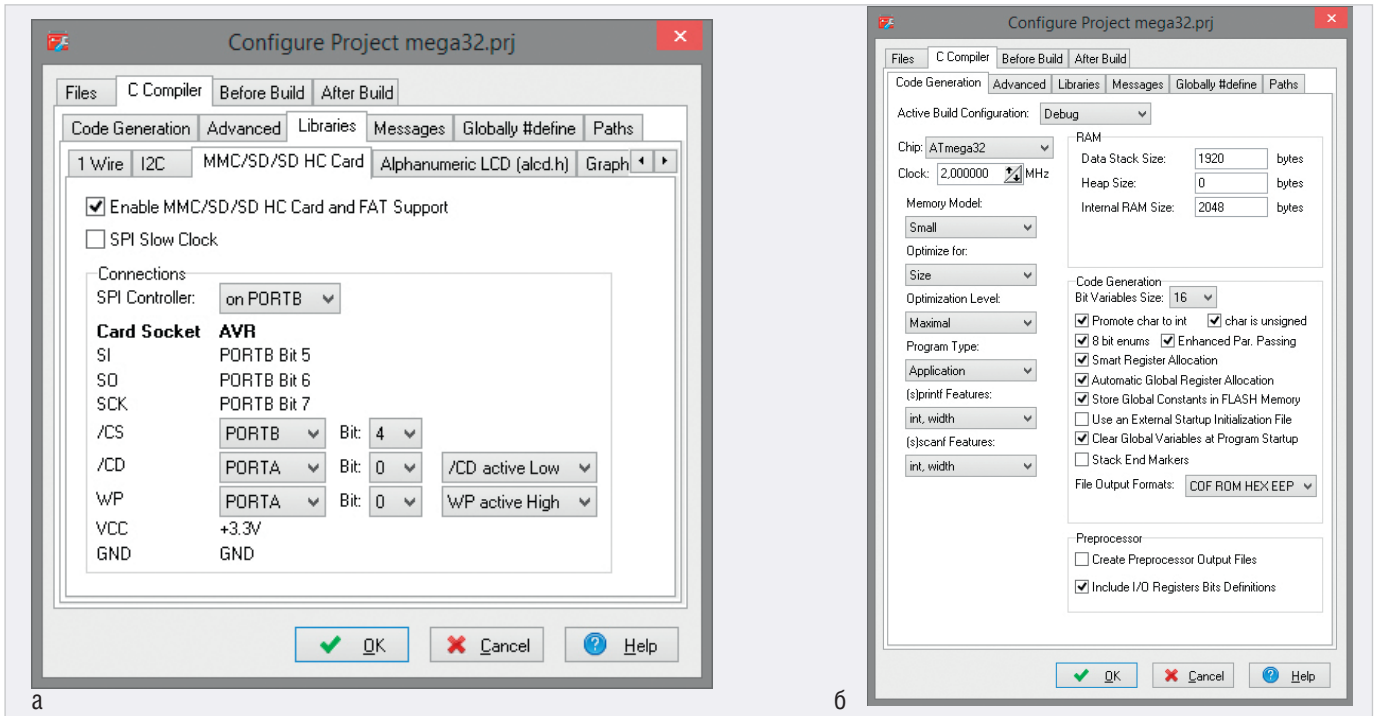
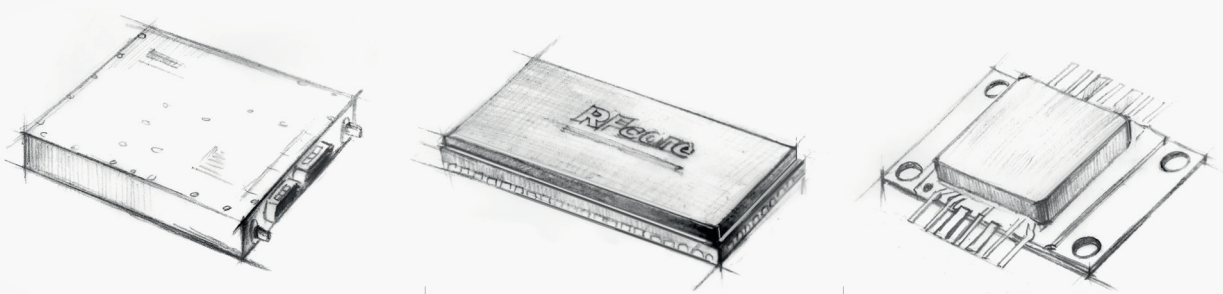


Рис. 5. Окно Configure Project: вкладка MMC/SD/SD HC Card (а), вкладка Code Generation (б)

лов, записи и чтения данных из внешней памяти библиотеки ff.h. Для этого создадим с помощью программы WinImage образ карты памяти (создание образа подробно рассмотрено в [1]), добавим в образ файл с расширением *.txt, который содержит блок текстовых данных, сохраним образ с расширением .ima в папке FAT16 в каталоге с проектом Proteus. Введём путь к файлу образа и его имя с расширением в поле Card Image File в окне свойств карты памяти (см. рис. 6). На языке С с применением функций библиотеки ff.h CodeVisionAVR напишем программу инициализации

RF-core
НОВЫЕ МОЩНОСТИ — НОВЫЕ ВОЗМОЖНОСТИ

СВЧ-усилители мощности
• Диапазон частот: от HF до Ku
• Выходная мощность: 2...1000 Вт
• Типовое усиление: 25...65 дБ
• Рабочее напряжение: 28, 40 В
Многофункциональные CMOS MMIC
• Диапазон частот: S, C, X, Ku
• Выходная мощность: до 15 Вт
• Исполнение: QFN-корпус
GaN и GaAs MMIC
• Диапазон частот: 2...18 ГГц
• Выходная мощность: до 12 Вт
• Типовое усиление: 10...23 дБ
• Исполнение: QFN-корпус/кристалл
ProChip powered by ProSoft
ОФИЦИАЛЬНЫЙДИСТРИБЬЮТОР
АКТИВНЫЙ КОМПОНЕНТ ВАШЕГО БИЗНЕСА
(495) 232-2522 ▪ INFO@PROCHIP.RU ▪ WWW.PROCHIP.RU

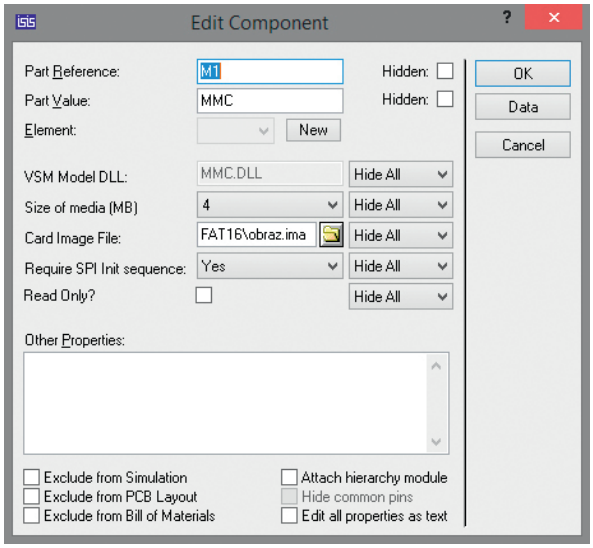


Рис. 6. Окно свойств карты памяти

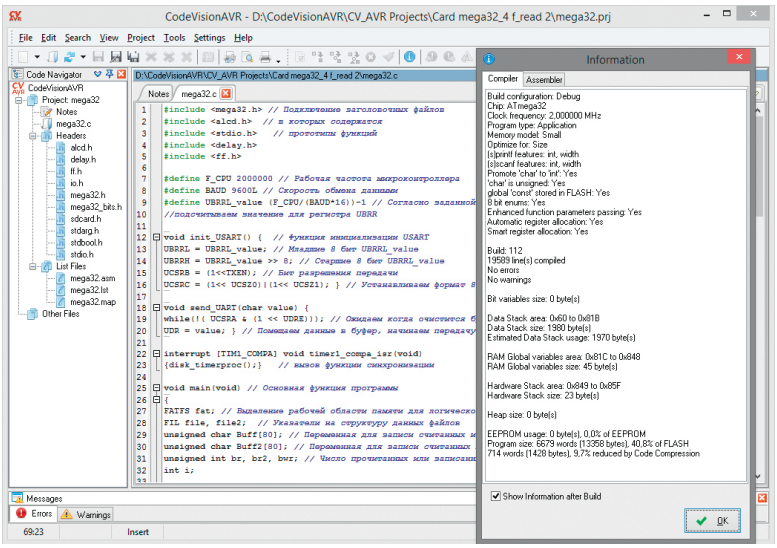


Рис. 7. Результат компиляции программного кода в CodeVisionAVR

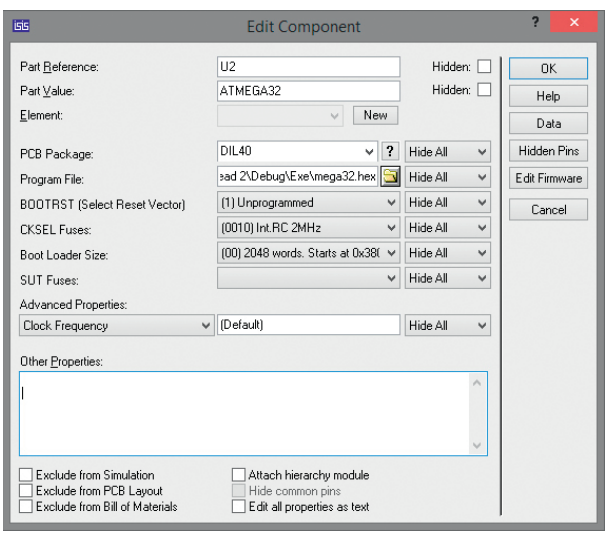


Рис. 8. Окно свойств микроконтроллера ATmega32

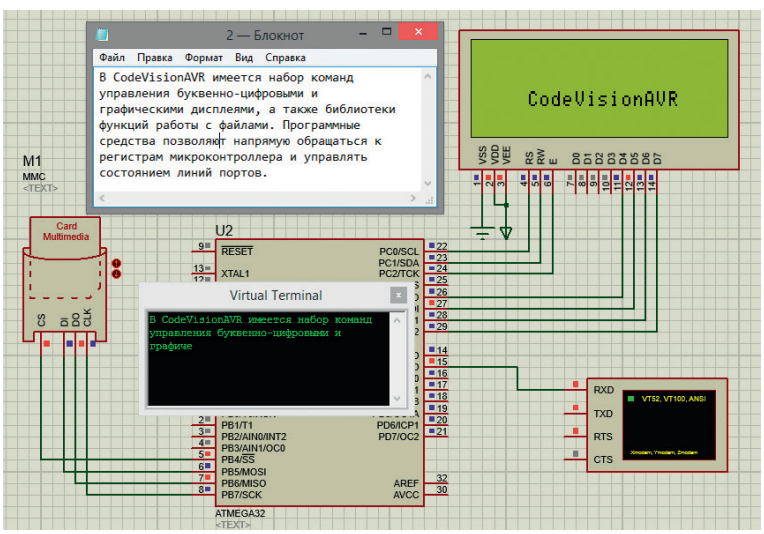


Рис. 9. Результат работы программы записи и чтения данных из внешней памяти, образ которой создан с помощью WinImage и сохранен в формате .ima

микроконтроллера. Текст программы приведён на листинге 1.

Введём текст программы в окне кода CodeVisionAVR и запустим компиляцию (см. рис. 7), по окончании которой будет создан .hex-файл для записи в микроконтроллер. Перейдём в Proteus и в окне свойств микросхемы ATmega32 укажем путь к файлу прошивки на диске компьютера (см. рис. 8). Результат симуляции схемы в Proteus представлен на рис. 9.

После запуска моделирования выполняется функция f_mount(0, &fat), которая выделяет область памяти нужного объёма для работы с картой памяти. Функция f_open(&file,«0:/1.txt», FA_OPEN_EXISTING | FA_READ) открывает уже размещённый на карте файл 1.txt для чтения из него с помощью функции f_read(&file, Buff, 80, &br) фрагмента данных размером 80 байт. После закрытия файла 1.txt и паузы длительностью 50 мс с помощью функции f_open(&file2,»0:/2.txt», FA_CREATE_ALWAYS | FA_WRITE) происходит создание на карте памяти ещё одного текстового файла 2.txt для записи в него функцией f_write(&file2, Buff, 80, &bwr) 80 байт данных, считанных из уже имеющегося на карте файла 1.txt (см. рис. 10). После закрытия и паузы функцией f_open(&file2,»0:/2.txt», FA_READ) файл 2.txt открывается повторно для чтения из него с помощью функции f_read(&file2, Buff2, 80, &br2) 80 байт данных и их вывода посимвольно в цикле for на экран терминала. Затем на экран дисплея выполняется посимвольный вывод в цикле for 13 символов из файла 2.txt.

Обмен информацией между микроконтроллером и картой памяти выполняется через линии PB4...PB7 микроконтроллера. Для работы с алфавитно-цифровым дисплеем задейство-

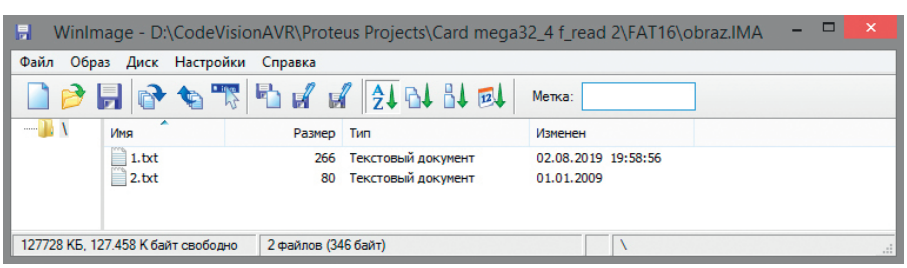


Рис. 10. Образ .ima, открытый в программе WinImage после добавления ещё одного файла с расширением *.txt

Листинг 1

```
#include <mega32.h> // Подключение заголовочных файлов
#include <alcd.h> // в которых содержатся
#include <stdio.h> // прототипы функций
#include <delay.h>
#include <ff.h>

#define F_CPU 2000000 // Рабочая частота микроконтроллера
#define BAUD 9600L // Скорость обмена данными
#define UBRRL_value (F_CPU/(BAUD*16))-1 // Согласно заданной скорости
// подсчитываем значение для регистра UBRR

void init_USART() { // Функция инициализации USART
UBRRL = UBRRL_value; // Младшие 8 бит UBRRL_value
UBRRH = UBRRL_value >> 8; // Старшие 8 бит UBRRL_value
UCSRB = (1<<TXEN); // Бит разрешения передачи
UCSRC = (1<< UCSZ0)|(1<< UCSZ1); } // Устанавливаем формат 8 бит данных

void send_UART(char value) {
while(!( UCSRA & (1 << UDRE))); // Ожидаем, когда очистится буфер передачи
UDR = value; } // Помещаем данные в буфер, начинаем передачу

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{disk_timerproc();} // Вызов функции синхронизации

void main(void)
{
FATFS fat; // Выделение рабочей области памяти для логического диска
FIL file, file2; // Указатели на структуру данных файлов
unsigned char Buff[80]; // Переменная для записи считанных из файла 1.txt данных
unsigned char Buff2[80]; // Переменная для записи считанных из файла 2.txt данных
unsigned int br, br2, bwr; // Число прочитанных или записанных байтов
int i;

// Инициализация портов микроконтроллера

// Port B
DDRB=(1<<DDB7) | (0<<DDB6) | (1<<DDB5) | (1<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) |
(0<<PORTB0);

// Port D, Port C
DDRD=DDRC=0xff;
PORTD=PORTC=0x00;

// Инициализация таймера
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x4E;
TIMSK=0x10;

lcd_init(20); // Инициализация дисплея
#asm("sei")
delay_ms(200);

init_USART(); // Инициализация USART
delay_ms(200);

f_mount(0, &fat); // Выделение рабочей области памяти для логического раздела
f_open(&file,"0:/1.txt", FA_OPEN_EXISTING | FA_READ); // Открываем файл 1.txt только для чтения
f_read(&file, Buff, 80, &br); // Читаем в переменную Buff 80 символов с начала файла
f_close(&file); // Закрываем файл 1.txt
delay_ms(50); // Пауза длительностью 50 мс

f_open(&file2,"0:/2.txt", FA_CREATE_ALWAYS | FA_WRITE ); // Создаем на карте памяти
// файл 2.txt для записи
f_write(&file2, Buff, 80, &bwr); // Записывем в файл 2.txt данные из файла 1.txt
f_close(&file2); // Закрываем файл
delay_ms(50); // Пауза длительностью 50 мс

f_open(&file2,"0:/2.txt", FA_READ); // Открываем файл 2.txt для чтения
f_read(&file2, Buff2, 80, &br2); // Читаем в переменную Buff2 80 символов с начала файла
f_close(&file2); // Закрываем файл

for (i=0;i<br2;i++)
{send_UART(Buff2[i]);} // Вывод считанной информации из созданного
// на карте памяти файла на экран терминала

lcd_gotoxy(4,2); // Установка курсора в четвертую позицию второй строки дисплея
for (i=2;i<15;i++)
{lcd_putchar(Buff2[i]);}} // Вывод на экран дисплея 13 символов, считанных из файла 2.txt карты памяти
```

ваны линии PC0...PC2 и PC4...PC7, линия PD1 микроконтроллера используется для последовательного вывода информации на экран терминала.

В представленном примере было выполнено чтение фиксированного числа байт информации из размещённого на карте памяти текстового файла. Для чтения всего файла удобно применить функцию feof(FIL *fp), которая во время доступа к файлу определяет, был ли достигнут его конец, и в этом случае возвращает ненулевое значение (1). В противном случае функция возвращает ноль (0). Параметр функции fp – это указатель на структуру данных файла. Размер файла в байтах определяют с помощью функции f_size(FIL *fp), реализованной в виде макроса #define f_size(fp) ((fp)->fsize), который определяют вместе с заголовочными файлами в начале кода программы:

```
#include <mega32.h>
#include <alcd.h>
#include <stdio.h>
#include <delay.h>
#include <ff.h>
```

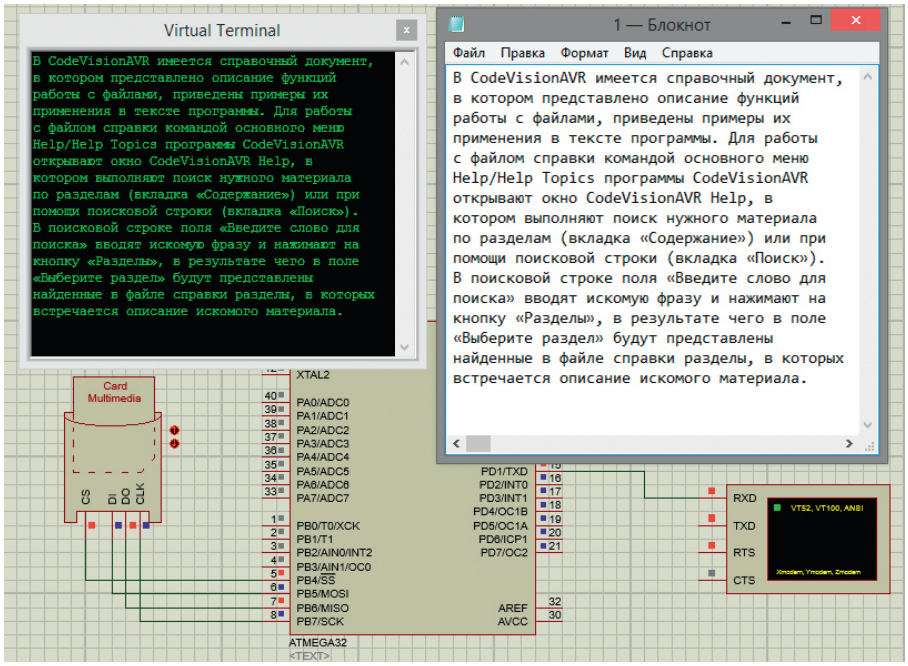


Рис. 11. Вывод на экран терминала всей записанной в файл 1.txt текстовой информации

```
#define f_size(fp) ((fp)->fsize)
```

Ниже представлен фрагмент основной функции программы инициализации, в котором даны указания микроконтроллеру смонтировать карту памяти и открыть для чтения размещённый на ней файл 1.txt. Функция (!feof(&file)) используется для определения достижения конца файла. Если символ завершения файла прочитан, то функция возвращает ненулевое значение, и цикл while завершается. После чего происходит закрытие файла и вывод в цикле for считанной из него информации на экран терминала.

Фрагмент основной функции программы инициализации микроконтроллера:

```
f_mount(0, &fat); // Выделение
рабочей области памяти для логиче-
ского раздела
f_open(&file,"0:/1.txt", FA_OPEN_
EXISTING | FA_READ); // Открываем
файл 1.txt
// только для чтения
while(!feof(&file)) { // Пока не
достигнут конец файла
f_read(&file, Buff, f_size(&file),
&br); // Считываем информацию
// и записываем в переменную Buff
}
f_close(&file); // Закрываем файл
1.txt
delay_ms(50); // Пауза длительно-
стью 50 мс
for (i=0;i<br;i++)
{send_UART(Buff[i]);} // Вывод
считанной информации на экран тер-
минала
```

В результате выполнения программы в Proteus на экран терминала выводится вся записанная в файле 1.txt текстовая информация (см. рис. 11).

Полезной при чтении/записи файла может быть функция f_lseek(FIL* fp, unsigned long ofs), которая перемещает указатель чтения/записи внутри файла, ранее открытого функцией f_open. Параметр fp функции указывает на структуру данных файла, которая должна быть предварительно инициализирована вызовом функции f_open. Параметр ofs задает смещение указателя чтения/записи файла на число байтов с начала файла. В режиме записи эта функция может использоваться для расширения размера файла путём перемещения указателя чтения/записи файла за конец имеющихся данных. При успешном выполнении значение элемента fptr структуры FIL, на которое указывает fp, должно быть проверено, чтобы убедиться в том, что указатель чтения/записи продвинулся в правильное положение и не произошло переполнение диска. В режиме чтения попытка продвинуть указатель чтения/записи файла за конец имеющихся данных ограничится установкой его в положение конца файла. В этом случае элемент fptr структуры FIL, на который указывает fp, будет содержать размер файла в байтах.

Ниже представлен фрагмент основной функции программы инициализации, в котором даны указания микроконтроллеру смонтировать карту памяти и открыть для чтения размещённый на ней файл 1.txt, затем с помощью функции f_lseek(&file, 120) сместить указатель чтения файла на 120 байтов с начала файла и с помощью функции f_read(&file, Buff, f_size(&file), &br) выполнить чтение оставшегося фрагмента файла. После чего происходит закрытие файла и вывод в цикле for считанной из него информации на экран терминала. Для определения достижения конца файла, как и в предыдущем примере, используется функция (!feof(&file)). Фрагмент основной функции программы инициализации микроконтроллера:

```
f_mount(0, &fat); // Выделение
рабочей области памяти для логиче-
ского раздела
f_open(&file,"0:/1.txt", FA_OPEN_
EXISTING | FA_READ); // Открываем
файл 1.txt
// только для чтения
f_lseek(&file, 120); // Смещение
указателя чтения файла на 120 бай-
тов с начала файла
while(!feof(&file)) { // Пока не
достигнут конец файла
f_read(&file, Buff, f_size(&file),
&br); // Считываем информацию
// и записываем в переменную Buff
}
f_close(&file); // Закрываем файл
1.txt
delay_ms(50); // Пауза длительно-
стью 50 мс
for (i=0;i<br;i++)
{send_UART(Buff[i]);} // Вывод
считанной информации на экран тер-
минала
```

В результате выполнения программы в Proteus происходит чтение из файла 1.txt карты памяти текстовых данных, начиная со 120 байта с начала файла, и их вывод на экран терминала (см. рис. 12).

Применение функций библиотек ff.h и glcd.h для чтения данных из внешней памяти и их отображения на экране графического дисплея

Используя микроконтроллер ATmega32, можно организовать одновременный вывод текстовых данных с карты памяти на экран алфавитно-цифрового и графического дисплеев. Для работы с графическим дисплеем в CodeVisionAVR имеются утилита формирования программного кода изображения – LCD Vision и библиотека glcd.h, которая содержит функции вывода текстовой и графической информации на экран дисплея, среди которых:

- glcd_outtext(char *str) – функция вывода текстовой строки в текущую

позицию графического экрана. Параметр str – это указатель на переменную, в которой хранятся текстовые данные;

- glcd_outtextxy(GLCDX_t x, GLCDY_t y, char *str) – функция вывода текстовой строки str, начиная с точки экрана с координатами *x*, *y*;
- glcd_putchar(char c) – функция вывода одиночного символа c в текущую позицию экрана;
- glcd_putcharxy(GLCDX_t x, GLCDY_t y, char c) – функция вывода одиночного символа c, начиная с точки экрана с координатами *x*, *y*;
- glcd_clear() – функция очистки экрана и установки курсора в позицию 0, 0;
- glcd_putimage(GLCDX_t left, GLCDY_t top, unsigned char *pimg, GLCDBLOCKMODE_t mode) – функция вывода на экран изображения, верхний левый угол которого будет размещён в точке с координатами left, top, где *pimg – указатель на переменную, в которой хранятся графические данные, mode – режим отображения изображения на экране. Значения аргумента mode: GLCD_PUTCOPY – перезапись отображаемого на экране изображения, GLCD_PUTTP – перезапись отображаемого на экране изображения в прозрачном режиме, когда для отображения используется цвет фона, GLCD_PUTXOR – исключающее ИЛИ с уже имеющимся на экране изображением, GLCD_PUTOR – объединяющее ИЛИ с уже имеющимся на экране изображением, GLCD_PUTNOT – вывод изображения с инверсией, GLCD_PUTAND – логическое И с уже имеющимся на экране изображением;
- glcd_getimage(GLCDX_t left, GLCDY_t top, GLCDX_t width, GLCDY_t height,

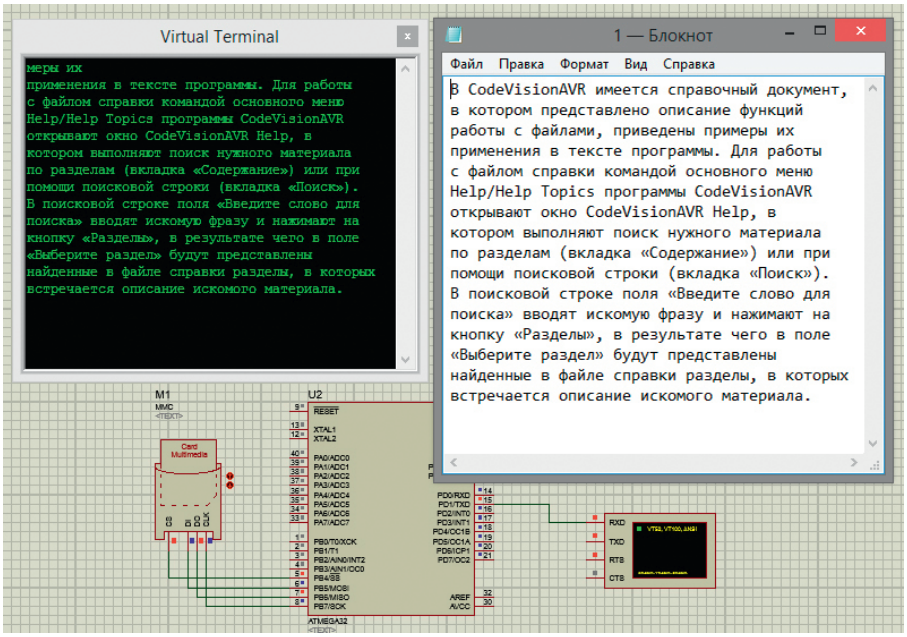


Рис. 12. Результат применения функции f_lseek(&file, 120) – чтение из файла 1.txt карты памяти текстовых данных, начиная со 120 байта с начала файла, и их вывод на экран терминала

АО «ВЗПП-С»
Тел/Факс:(473) 223-69-51
E-mail: market@vzpp-s.ru
www.vzpp-s.ru
НОВЫЕ СЕРИЙНЫЕ ИЗДЕЛИЯ
Изделия выпускаются в 20 различных корпусах
1. Быстровосстанавливающиеся диоды и диодные сборки серии 2ДВ102, 103, 104, 105(13 типономиналов) (200 ÷ 600 В, 1 ÷ 35 А, 25 ÷ 60 нс)
2. Однофазные мосты серии 2МД147, 148, 149 (8 типономиналов) (200 ÷ 600 В, 0,5 ÷ 25 А, 1 ÷ 1,1 В)
3. Диоды Шоттки и диодные сборки (23 типономинала) (5 ÷ 200 В, 0,001 ÷ 70(2×35) А, 0,25 ÷ 1,18 В)
4. Двухканальные драйверы серии 1347 (6 типономиналов) (6 ÷ 20 В, 1,5 А/-1,5 А, 70/75 нс)
5. ШИМ-контроллеры серии 5319 (4 типономинала) (28 В, ± 0,9 А, 500 кГц)
6. МКМ серии 3005 (4 типономинала):
- Двухполярный источник тока;
- Преобразователь напряжения на датчике тока;
- Преобразователи входных дискретных сигналов;
- МКМ управления источником питания.
Реклама

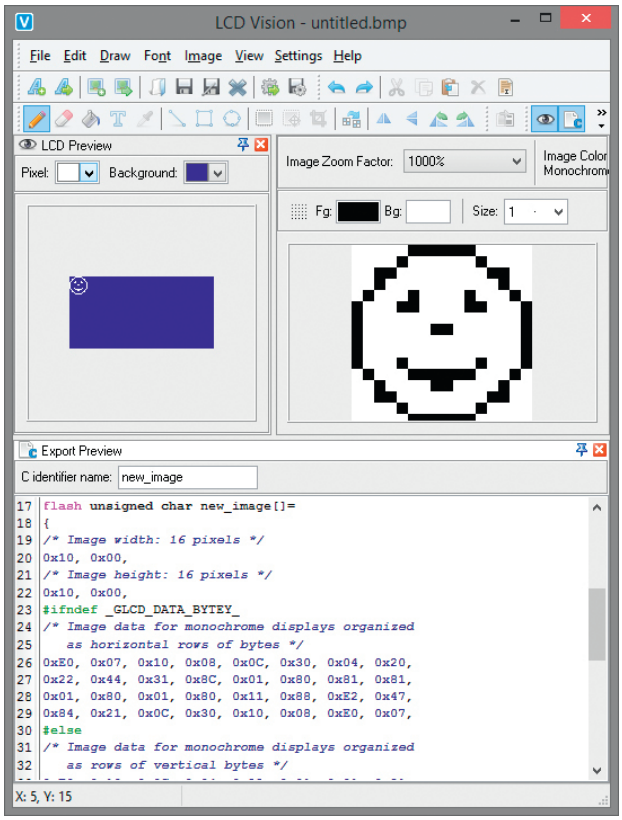


Рис. 13. Рисунок размером 16×16 пикселей и его код в окне Export Preview утилиты LCD Vision

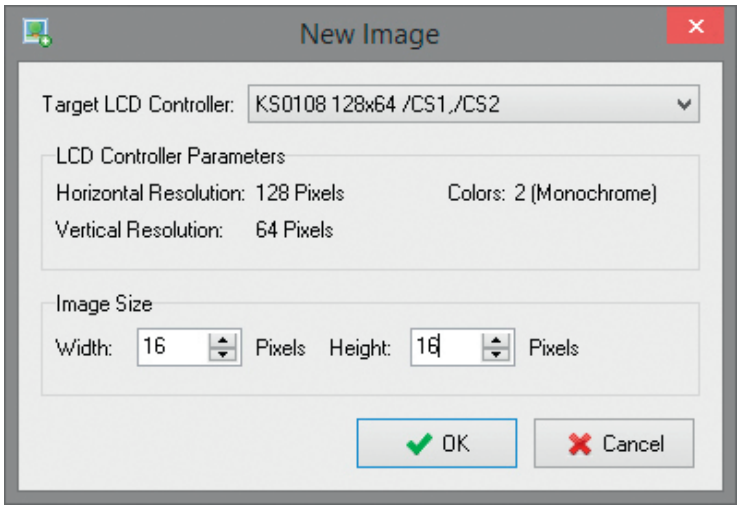


Рис. 14. Окно New Image утилиты LCD Vision

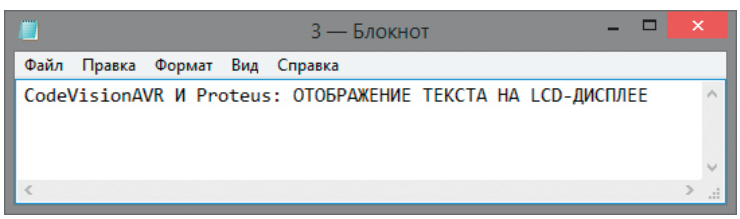


Рис. 15. Текстовая информация, записанная в размещенном на карте памяти файле 3.txt

unsigned char *pimg) – функция копирования с экрана фрагмента изображения шириной width и высотой height, верхний левый угол которого размещён в точке с координатами left, top, где *pimg – указатель на переменную, в которой будут храниться скопированные графические данные;

- glcd_setlinestyle(unsigned char thickness, unsigned char bit_pattern) – функция устанавливает стиль рисования линий на экране графического дисплея, где thickness – толщина линий в пикселях, bit_pattern – стиль линий. Значения аргумента bit_pattern: GLCD_LINE_SOLID – сплошная линия, GLCD_LINE_DOT_SMALL – точечная линия, GLCD_LINE_DOT_LARGE – жирная точечная линия;
- glcd_setlinethick(unsigned char thickness) – функция устанавливает толщину линий thickness в пикселях;
- glcd_line(GLCDX_t x0, GLCDY_t y0, GLCDX_t x1, GLCDY_t y1) – функция отрисовывает на экране линию, где $x0$, $y0$ – координаты ее начала, $x1$, $y1$ – координаты конца;
- glcd_rectangle(GLCDX_t left, GLCDY_t top, GLCDX_t right, GLCDY_t bottom) – функция отрисовывает на экране прямоугольник, где left, top – координаты его левого верхнего угла, right bottom – координаты правого нижнего угла;
- glcd_rectround(GLCDX_t left, GLCDY_t top, GLCDDX_t width, GLCDDY_t height, GLCDRAD_t radius) – функция отрисовывает на экране прямоугольник, где left, top – координаты его левого верхнего угла, right bottom – координаты правого нижнего угла, radius – радиус скругления углов прямоугольника;
- glcd_circle(GLCDX_t x, GLCDY_t y, GLCDRAD_t radius) – функция отрисовывает окружность, где x, y – координаты её центра, radius – радиус окружности;
- glcd_putpixel(GLCDX_t x, GLCDY_t y, GLCDCOL_t color) – функция закрашивает на экране пиксель цветом color с координатами x, y. Для монохромных дисплеев параметр color может принимать одно из двух значений: 0xffff (черный), 0x0000 (цвет фона экрана);
- glcd_init() – инициализация графического дисплея.

Для построения изображения на экране графического дисплея используется простая система координат. Отсчёт начинается от верхнего левого угла экрана, который имеет координаты 0,0. Значение x увеличивается слева направо, значение y увеличивается сверху вниз.

Для вывода изображения на экран дисплея его необходимо конвертировать в массив данных, для чего удобно применить утилиту LCD Vision (см. рис. 13), которую используют для создания изображений и экспорта их кода в CodeVisionAVR. Изображения могут быть импортированы в LCD Vision из популярных графических форматов (таких как BMP, JPG, GIF, PNG, ICO, WMF, EMF) для последующей генерации их кода или созданы с нуля, для чего применяют команду основного меню утилиты File/New Image. В результате выполнения команды открывается окно New Image (см. рис. 14), в котором в меню Target LCD Controller из выпадающего списка выбирают целевой контроллер графического дисплея с нужными характеристиками. В нашем примере это: тип контроллера – KS0108, разрешение экрана – 128×64, инверсия входов активации сегментов дисплея – /CS1, /CS2). В поле Image Size указывают размер изображения в пикселях (в нашем примере ширина Width и высота Height изображения 16 пикселей). Затем нажимают на кнопку OK.

Далее в LCD Vision в поле Image Zoom Factor выбирают масштаб окна рабочего поля (в нашем примере установим значение 1000% для максимального приближения рисунка), в котором с помощью инструментов: Set pixel (установить пиксель), Erase pixel (стереть пиксель), Fill area (заполнить область), Draw line (нарисовать линию), Draw

Свобода проектирования

DeltaDesign

В состав Delta Design, обеспечивающей сквозной цикл проектирования печатных плат, входят модули:

- Менеджер библиотек
- Схемотехнический редактор
- Схемотехническое моделирование
- HDL-симулятор
- Редактор правил
- Редактор печатных плат
- Топологический редактор плат TopoR
- Коллективная работа для предприятий

EREMEX +7 (495) 232-18-64 • sales@eremex.ru • www.eremex.ru

Реклама

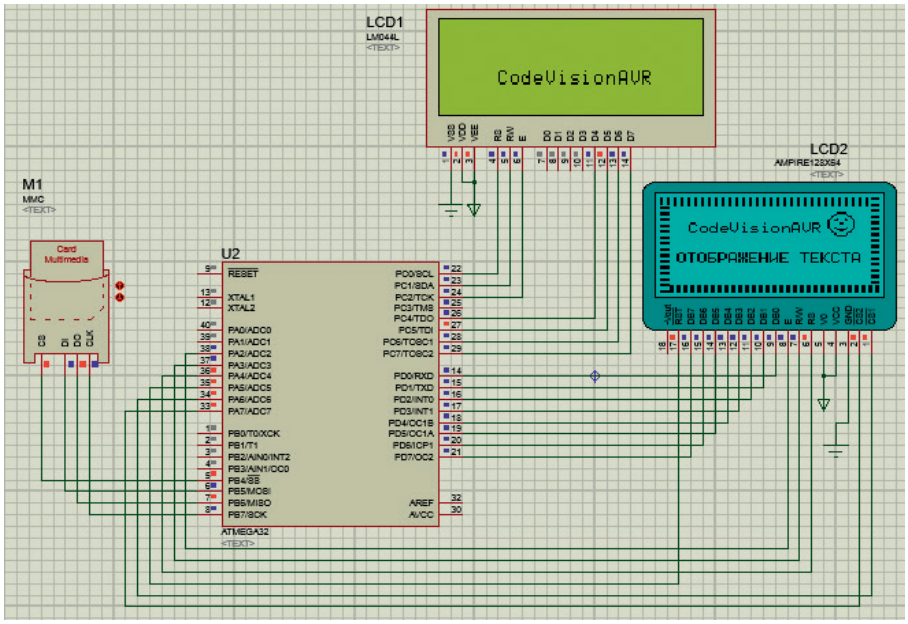


Рис. 16. Вывод с помощью функций библиотек ff.h и glcd.h текстовой и графической информации на экран алфавитно-цифрового и графического дисплея

rectangle (нарисовать прямоугольник), Draw ellipse or circle (нарисовать окружность) панели Draw создают рисунок. Зажав левую клавишу мыши, отрисовывают изображение, генерируют его код нажатием пиктограммы Export панели инструментов File. Запуск утилиты LCD Vision выполняют из программы CodeVisionAVR нажатием пиктограммы LCD Vision font and image editor/converter панели инструментов Tools (панель открывают командой View/Toolbars/Tools основного меню).

Для чтения данных с карты памяти удобно применить функции библиотеки ff.h, описание которых было рассмотрено в [1]. В качестве примера сформируем на экране дисплея AMPIRE128×64 с помощью функций библиотеки glcd.h графику, а также выведем текстовую информацию, хранящуюся в файле, размещённом на карте памяти. На экран алфавитно-цифрового дисплея выведем фрагмент данных, считанный из этого же файла. С этой целью создадим с помощью программы WinImage образ карты памяти, добавим в образ файл с расширением *.txt, который содержит блок текстовых данных (см. рис. 15), сохраним образ в папке FAT16 в каталоге с проектом Proteus с расширением .ima и введём вручную путь к файлу образа и его имя в поле Card Image File в окне свойств карты памяти. Для чтения данных из размещённого во внешней памяти файла и их отображения на экране графического и алфавитно-цифрового дисплея напишем программу инициализации микроконтроллера на языке С с применением стандартных функций CodeVisionAVR (см. листинг 2).

Введём текст программы в окне кода CodeVisionAVR и запустим компиляцию. После чего перейдём в Proteus и в окне свойств микросхемы ATmega32 укажем путь к файлу прошивки на диске компьютера. Запустим симуляцию собранной схемы, результат выполнения которой представлен на рис. 16, и проанализируем её работу.

После запуска моделирования программа микроконтроллера выполняет инициализацию алфавитно-цифрового (функция lcd_init(20)) и графического дисплея (функция glcd_init(&glcd_init_data)) и определение шрифта вывода текста (glcd_init_data.font=font5x7) на экран графического дисплея. После чего выполняется выделение области памяти нужного объёма для работы с картой памяти (функция f_mount(0, &fat)), установка стиля рисования линий на графическом дисплее (функция glcd_setlinestyle(6,GLCD_LINE_DOT_LARGE)) и отрисовка на его экране прямоугольной рамки (функция glcd_rectangle(0,0,128,64)).

Функция f_open(&file,«0:/3.txt», FA_OPEN_EXISTING | FA_READ) открывает уже размещённый на карте памяти файл 3.txt для чтения из него с помощью функции f_read(&file, Text, 13, &br) фрагмента данных размером 13 байт. Вывод считанных символов на экран графического дисплея выполняется с помощью функции glcd_outtextxy(17,17,Text), начиная с позиции x = 17, y = 17. Функция lcd_gotoxy(4,2) определяет позицию курсора на экране алфавитно-цифрового дисплея, начиная с которой в цикле for выполняется посимвольный вывод данных, считанных из файла 3.txt карты памяти.

Далее выполняется смещение указателя чтения на 25 байт с начала файла (функция f_lseek(&file, 25)), чтение 18 символов в переменную Text (функция f_read(&file,Text, 18, &br)) и вывод считанных символов на экран графического дисплея, начиная с позиции x = 10, y = 37 (функция glcd_outtextxy(10,37,Text)).

После закрытия размещённого на карте памяти текстового файла (функция f_close(&file)) и паузы длительностью 50 мс (функция delay_ms(50)) с помощью функции glcd_putimage(98,10,img,GLCD_PUTCOPY) выполняется вывод изображения, код которого хранится в переменной img, на экран графического дисплея, начиная с позиции $x = 98, y = 10$ (см. рис. 17а).

Рассмотрим ещё один пример, в котором выполним захват данных с экрана графического дисплея и вывод изображения, код которого записан в текстовом файле, размещённом на карте памяти. С этой целью внесем в представленный листинг изменения, дополнив его следующим фрагментом кода:

```
glcd_getimage(98, 10, 16, 16,
img2); // Запись фрагмента графи-
ческих данных
// с экрана в переменную img2
f_open(&file2,"0:/9.txt", FA_
CREATE_ALWAYS | FA_WRITE ); //
Создание на карте
// памяти файла 9.txt для записи
f_write(&file2, img2,
strlen(img2), &bwr); // Запись в
файл 9.txt данных
// считанных с экрана графическо-
го дисплея
f_close(&file2); // Закрываем файл
glcd_clear(); // Очищаем экран
графического дисплея
f_open(&file2,"0:/9.txt", FA_OPEN_
EXISTING | FA_READ); // Открываем
на карте памяти
// файл 9.txt только для чтения
f_read(&file2, img3, f_
size(&file2), &br2); // Считываем
данные в переменную img3
glcd_putimage(43,30,img3,GLCD_
PUTCOPY); // Вывод изображения,
код которого записан
// в переменной img3, на экран
графического дисплея с позиции x =
43, y = 30
```

Листинг 2

```
#include <mega32.h> // Подключение заголовочных файлов
#include <alcd.h> // в которых содержатся
#include <stdio.h> // прототипы функций
#include <delay.h>
#include <ff.h>
#include <glcd.h>
#include <font5x7rus.h>

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{disk_timerproc();} // Вызов функции синхронизации

void main(void) // Основная функция программы
{
FATFS fat; // Выделение рабочей области памяти для логического диска
FIL file; // Указатель на структуру данных файла
unsigned char Text[58],img[]={
0x10,0x00, // Размер изображения по горизонтали 16 пикселей
0x10,0x00, // Размер изображения по вертикали 16 пикселей
#ifndef _GLCD_DATA_BYTEY_
/* Код изображения, организованный в виде горизонтальных строк байтов */
0xE0, 0x07, 0x10, 0x08, 0x0C, 0x30, 0x04, 0x20, 0x22, 0x44, 0x31, 0x8C, 0x01, 0x80, 0x81, 0x81,
0x01, 0x80, 0x01, 0x80, 0x11, 0x88, 0xE2, 0x47, 0x84, 0x21, 0x0C, 0x30, 0x10, 0x08, 0xE0, 0x07,
#else
/* Код изображения, организованный в виде вертикальных строк байтов */
0xE0, 0x10, 0x0C, 0x04, 0x22, 0x31, 0x01, 0x81, 0x81, 0x01, 0x31, 0x22, 0x04, 0x0C, 0x10, 0xE0,
0x07, 0x08, 0x30, 0x20, 0x44, 0x88, 0x88, 0x98, 0x98, 0x88, 0x88, 0x44, 0x20, 0x30, 0x08, 0x07,
#endif
};

unsigned int br; // Число прочитанных байтов
int i;
GLCDINIT_t glcd_init_data; // Переменная для хранения данных контроллера
// графического дисплея

// Инициализация портов микроконтроллера
// Port B
DDRB=(1<<DDB7) | (0<<DDB6) | (1<<DDB5) | (1<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (0<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) |
(0<<PORTB0);

// Port D, Port C
DDRD=DDRC=0xff;
PORTD=PORTC=0x00;

// Инициализация таймера
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x4E;
TIMSK=0x10;

lcd_init(20); // Инициализация алфавитно-цифрового дисплея
#asm("sei")
delay_ms(200); // Пауза длительностью 200 мс
glcd_init_data.font=font5x7; // Определение шрифта графического дисплея
glcd_init(&glcd_init_data); // Инициализация графического дисплея

f_mount(0, &fat); // Выделение рабочей области памяти для логического раздела
glcd_setlinestyle(6,GLCD_LINE_DOT_LARGE); // Установка стиля рисования линий на
// графическом дисплее
glcd_rectangle(0,0,128,64); // Отрисовка прямоугольной рамки
f_open(&file,"0:/3.txt", FA_OPEN_EXISTING | FA_READ); // Открываем на карте памяти
// файл 3.txt только для чтения
f_read(&file, Text, 13, &br); // Читаем с карты памяти в переменную Text
// 13 символов с начала файла 3.txt
glcd_outtextxy(17,17,Text); // Вывод на экран графического дисплея считанных символов
lcd_gotoxy(4,2); // Установка курсора в четвёртую позицию
// второй строки алфавитно-цифрового дисплея
for (i=0;i<13;i++)
{lcd_putchar(Text[i]);} // Посимвольный вывод на экран
// алфавитно-цифрового дисплея считанных из файла 3.txt карты памяти данных
f_lseek(&file, 25); // Смещение указателя чтения файла на 25 байтов с начала файла
f_read(&file,Text, 18, &br); // Чтение с карты памяти в переменную Text 18 символов
glcd_outtextxy(10,37,Text); // Вывод на экран графического дисплея считанных символов
f_close(&file); // Закрываем файл 3.txt
delay_ms(50); // Пауза длительностью 50 мс
glcd_putimage(98,10,img,GLCD_PUTCOPY); // Вывод изображения, код которого указан
// в переменной img, на экран графического дисплея с позиции x = 98, y = 10
delay_ms(50);
}
```

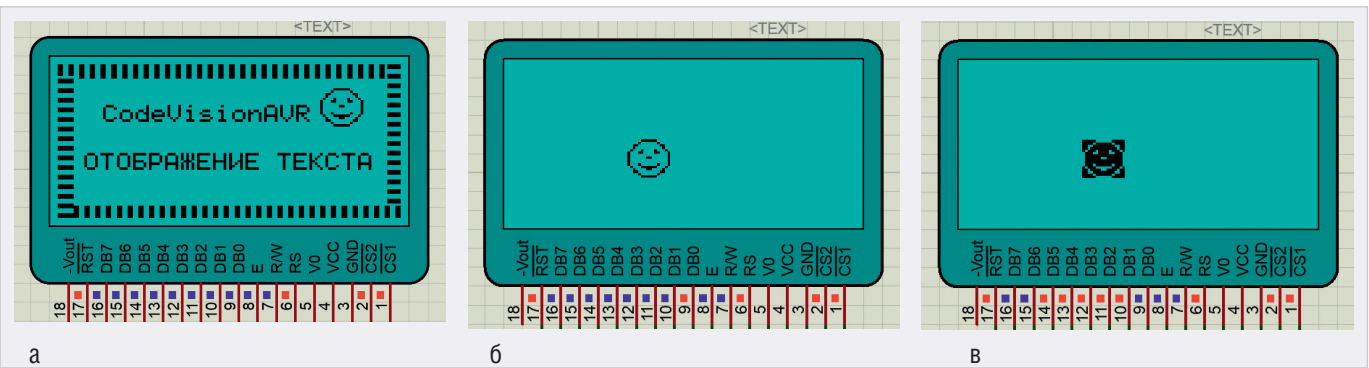


Рис. 17. Приближённый вид графического дисплея после вывода считанной из внешней памяти: текстовой информации (а), графической информации в режиме GLCD_PUTCOPY (б), графической информации в режиме GLCD_PUTNOT (в)

В результате выполнения программы с экрана графического дисплея был скопирован фрагмент изображения, размер которого и начальные координаты отображения на дисплее определяют параметры функции glcd_getimage(98, 10, 16, 16, img2), где img2 – переменная для хранения графической информации. После чего с помощью функции f_open(&file2,»0:/9.txt», FA_CREATE_ALWAYS | FA_WRITE) был создан новый файл на карте памяти, в который функцией f_write(&file2, img2, strlen(img2), &bwr) выполнена запись данных, скопированных с экрана графического дисплея. После очистки экрана дисплея (функция glcd_clear()) скопированный фрагмент графики выводится из размещённого на карте памяти файла (f_open(&file2,«0:/9.txt», FA_OPEN_EXISTING | FA_READ); f_read(&file2, img3, f_size(&file2), &br2)) на экран графического дисплея, начиная с позиции x = 43, y = 30 (glcd_putimage(43,30,img3,GLCD_PUTCOPY)).

Определение количества байтов для записи в файл 9.txt и чтения из файла в представленном фрагменте кода реализовано с помощью функций strlen(img2) и f_size(&file2), для применения которых в начале кода программы добавлены следующие строки:

```
#include <string.h>
#define f_size(fp) ((fp)->fsize)
```

и объявлены переменные:

```
FIL file2;
unsigned char img2[], img3[].
```

Результат работы программы представлен на рис. 17б. На рис. 17в показано отображение фрагмента рисунка в режиме GLCD_PUTNOT на экране графического дисплея.

Применение функций библиотеки ff.h для создания, переименования, удаления директорий и файлов на карте памяти

Библиотека ff.h программной среды CodeVisionAVR содержит ряд функций для управления файловой структурой внешней памяти, среди которых:

- f_rename(const char* path_old, const char* path_new) – функция переименования размещённых на карте памяти файлов или директорий, где path_old – указатель на старое имя объекта, path_new – указатель на новое имя объекта. Функция возвращает следующие значения:
  - FR_OK – успешное выполнение функции;
  - FR_NO_FILE – не удалось найти файл или директорию;
  - FR_NO_PATH – путь к файлу не существует;
  - FR_INVALID_NAME – неверное имя файла или директории;
  - FR_INVALID_DRIVE – неверный номер устройства;
  - FR_EXIST – создание файла/директории невозможно, так как файл/директория с таким именем уже существует;
  - FR_DENIED – файл или директория не могут быть созданы или перемещены по неизвестной причине;
  - FR_NOT_READY – доступ к карте памяти невозможен из-за отсутствия её подключения или по другой причине;
  - FR_WRITE_PROTECTED – носитель защищён от записи;
  - FR_DISK_ERR – ошибка доступа к карте памяти;
  - FR_INT_ERR – внутренняя ошибка карты памяти или файловой системы FAT;

Акционерное общество
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ
ГИРИКОНД

РАСШИРЕНИЕ ШКАЛЫ ТИПОНОМИНАЛОВ

Низковольтные многослойные керамические чип-конденсаторы
- Номинальное напряжение: 6,3 В - 500 В
- Номинальная емкость: 1пФ - 15 мкФ
- Габаритные размеры: от 0402 (1 0,5 мм)
- Группы ТСЕ: МП0; Н20; Н30
- Диапазон рабочих температур: - 60 С до +125 С

с освоением в серийном производстве

Керамические помехоподавляющие фильтры
- Номинальное напряжение: 32 - 1000 В
(теперь для групп ТСЕ: Н20; Н50 до 750 В)
- Номинальная емкость: 4,7пФ - 22 мкФ
- Номинальные токи: 10; 15; 25 А
- Группы ТСЕ: МП0; Н20; Н50; Н90
- Диапазон рабочих температур: - 60 С до +125 С
- Габаритные размеры корпуса: от 7,5 4,0 4,6 мм
(включая миниатюрные фильтры С, LC и Pi-типов с диаметром шайбы от 3 мм и резьбой от М3)

Рекомендуются в качестве отечественных аналогов при решении задач импортозамещения

Реклама
194223, Санкт-Петербург, ул. Курчатова, 10
Тел.: (812) 247-14-92 Факс: (812) 552-60-57
www.giricond.ru
E-mail: 333@giricond.ru

- FR_NOT_ENABLED – логический диск не был смонтирован с помощью функции f_mount;
- FR_NO_FILESYSTEM – на физическом носителе отсутствует корректный раздел FAT;

- f_mkdir (const char* path) – функция создания новой директории на карте памяти, где path – номер устройства и имя директории для создания. Функция возвращает следующие значения:
  - FR_OK – успешное выполнение функции;
  - FR_NO_PATH – указанный в переменной path путь для создания директории не найден;
  - FR_INVALID_NAME – неверное имя директории;
  - FR_INVALID_DRIVE – неверный номер устройства;
  - FR_EXIST – директория с указанным именем уже существует;
  - FR_DENIED – отсутствует свободное место на карте памяти;
  - FR_NOT_READY – доступ к карте памяти невозможен из-за отсутствия её подключения или по другой причине;
  - FR_WRITE_PROTECTED – создание директории невозможно, потому что носитель защищён от записи;
  - FR_DISK_ERR – ошибка доступа к карте памяти;
  - FR_INT_ERR – внутренняя ошибка карты памяти или файловой системы FAT;
  - FR_NOT_ENABLED – логический диск не был смонтирован с помощью функции f_mount;
  - FR_NO_FILESYSTEM – на физическом носителе отсутствует корректный раздел FAT;
- f_unlink(const char* path) – функция удаления файлов и директорий на карте памяти, где path – номер устройства и имя файла или директории. Функция возвращает следующие значения:
  - FR_OK – успешное выполнение функции;
  - FR_NO_FILE – указанный для удаления файл или директория не существуют;
  - FR_NO_PATH – указанный путь отсутствует;
  - FR_INVALID_NAME – неверное имя файла или директории;
  - FR_INVALID_DRIVE – неверный номер устройства;
  - FR_DENIED – файл или директория не могут быть удалены по одной из следующих причин: для файла/директории установлен атрибут «только для чтения», директория содержит файлы и папки;
  - FR_NOT_READY – доступ к карте памяти невозможен из-за отсутствия её подключения или по другой причине;
  - FR_WRITE_PROTECTED – удаление директории/файла невозможно, потому что носитель защищён от записи;
  - FR_DISK_ERR – функция не может быть выполнена из-за отсутствия доступа к карте памяти;
  - FR_INT_ERR – внутренняя ошибка карты памяти или файловой системы FAT;
  - FR_NOT_ENABLED – логический диск не был смонтирован с помощью функции f_mount;

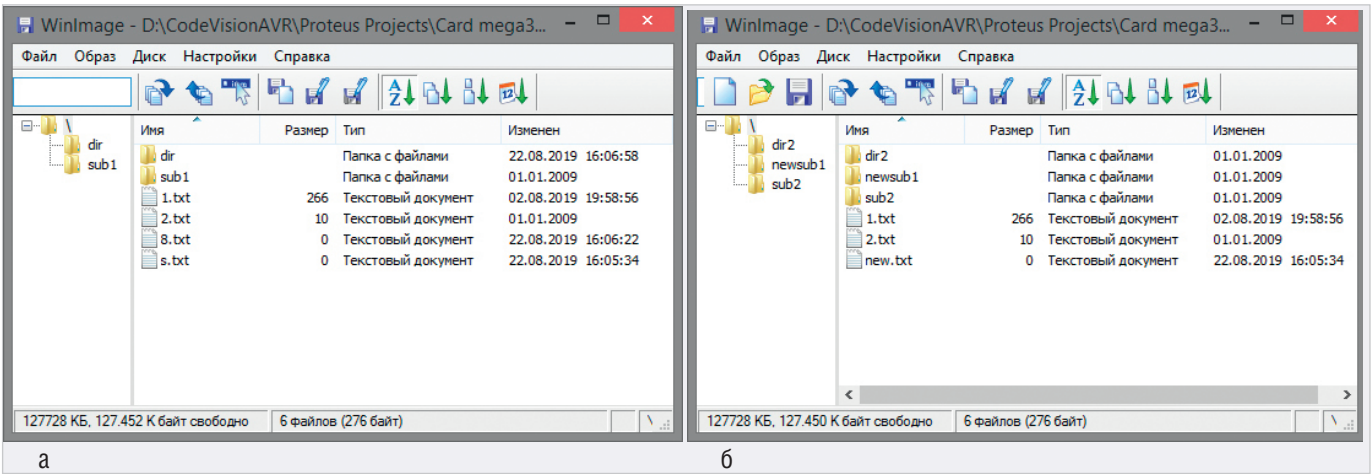


Рис. 18. Образ карты памяти: до (а) и после (б) применения функций создания, переименования и удаления файлов и папок

Листинг 3

```
#include <mega32.h> // Подключение заголовочных файлов
#include <ff.h>

interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{disk_timerproc();} // Вызов функции синхронизации

void main(void) // Основная функция программы
{
FATFS fat; // Выделение рабочей области памяти для логического диска

// Инициализация порта PB микроконтроллера
DDRB=(1<<DDB7) | (0<<DDB6) | (1<<DDB5) | (1<<DDB4) | (0<<DDB3) |
(0<<DDB2) | (0<<DDB1) | (0<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) |
(0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);

// Инициализация таймера
TCCR1A=0x00;
TCCR1B=0x0D;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1AH=0x00;
OCR1AL=0x4E;
TIMSK=0x10;
#asm("sei")

f_mount(0, &fat); // Выделение рабочей области памяти для логического
раздела
f_rename("s.txt", "new.txt"); // Переименование файла
f_rename("/sub1", "/newsub1"); // Переименование папки
f_mkdir("0:/sub2"); // Создание новой папки sub2
f_mkdir("0:/dir2"); // Создание новой папки dir2
f_unlink("0:/dir"); // Удаление папки dir
f_unlink("0:/8.txt"); // Удаление файла
}
```

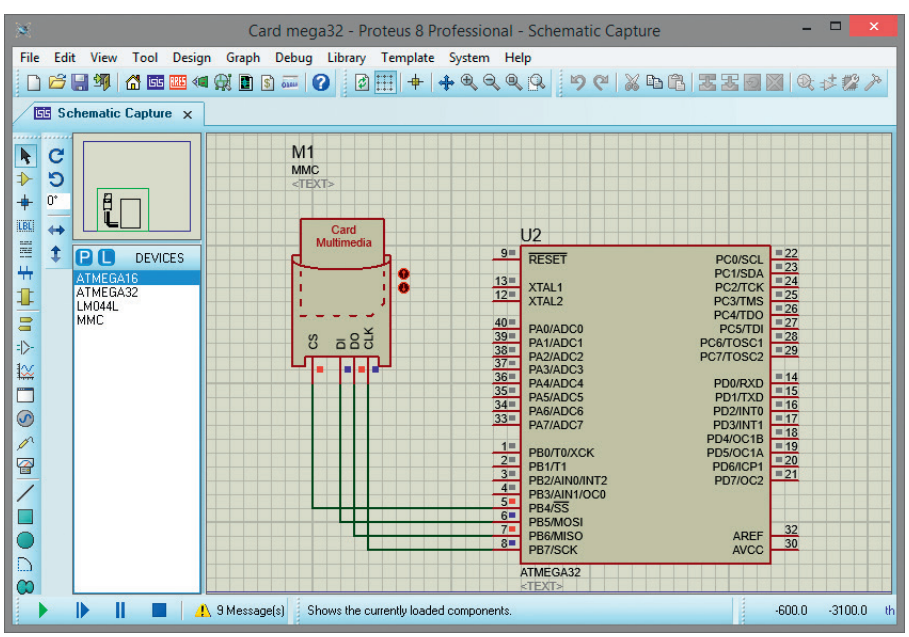


Рис. 19. Симуляция схемы управления файловой структурой карты памяти с помощью микроконтроллера ATmega32 в редакторе ISIS программы Proteus

– FR_NO_FILESYSTEM – на физическом носителе отсутствует корректный раздел FAT.

Перед выполнением функций переименования и удаления открытые объекты должны быть закрыты.

Рассмотрим работу с функциями на конкретном примере. Для этого создадим с помощью программы WinImage образ карты памяти, добавим в образ две папки dir и sub1 и текстовые файлы s.txt, 1.txt, 2.txt, 8.txt (см. рис. 18а). Сохраним образ в папке FAT16 в каталоге с проектом Proteus с расширением .ima и введём вручную путь к файлу образа и его имя в поле Card Image File в окне свойств карты памяти. Используя представленные функции, переименуем уже имеющийся на карте памяти файл s.txt и папку sub1, создадим две новые директории sub2 и dir2 и удалим файл 8.txt и папку dir (см. рис. 18б). Текст программы представлен на листинге 3.

Введём текст программы в окне кода CodeVisionAVR и запустим компиляцию. После чего перейдём в Proteus и в окне свойств микросхемы ATmega32 укажем путь к файлу прошивки на диске компьютера. Запустим симуляцию собранной схемы (см. рис. 19), в результате чего будет выполнено монтирование карты памяти, переименование текстового файла и папки, создание двух новых директорий, удаление директории и файла на карте памяти.

Литература

1. *Колесникова Т.* Проектирование схем микроэлектронных устройств с использованием внешней памяти в Proteus. Ч. 1 // Современная электроника. 2021. № 6.
2. ISIS Help, Labcenter Electronics, 2014.
3. CodeVisionAVR Help, HP InfoTech, 2014.
4. HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver). Hitachi, Ltd. 1998.

КУРС НА ИМПОРТОЗАМЕЩЕНИЕ

ПРОМЫШЛЕННЫЕ КОМПЬЮТЕРЫ ADVANTIX «БРУСНИКА» НА БАЗЕ ЦПУ «ЭЛЬБРУС»

РАЗРАБОТАНО И СДЕЛАНО В РОССИИ

- Разработано и произведено в России
- Отечественные процессоры «Эльбрус»
- Безвентиляторное исполнение
- Для критической инфраструктуры
- Фиксация кабеля питания
- Корпуса для установки в 19″ стойку
- Поддержка отечественных операционных систем
- Возможность заказных разработок

ProSoft® WWW.PROSOFT.RU
ОФИЦИАЛЬНЫЙ ДИСТРИБЬЮТОР

МОСКВА	(495) 234-0636	info@prosoft.ru
С.-ПЕТЕРБУРГ	(812) 448-0444	info@spb.prosoft.ru
АЛМА-АТА	(727) 321-8324	sales@kz.prosoft.ru
ВОЛГОГРАД	(8442) 391-000	volgograd@regionprof.ru
ВОРОНЕЖ	(473) 229-5281	voronezh@regionprof.ru
ЕКАТЕРИНБУРГ	(343) 356-5111	info@prosoftsystems.ru
	(912) 620-8050	ekaterinburg@regionprof.ru
КАЗАНЬ	(843) 203-6020	kazan@regionprof.ru
КРАСНОДАР	(861) 224-9513	krasnodar@regionprof.ru
Н. НОВГОРОД	(831) 261-3484	n.novgorod@regionprof.ru
НОВОСИБИРСК	(383) 335-7001	nsk@regionprof.ru
ОМСК	(3812) 286-521	omsk@regionprof.ru
ПЕНЗА	(8412) 49-4971	penza@regionprof.ru
ПЕРМЬ	(912) 059-0757	perm@regionprof.ru
САМАРА	(846) 277-9166	samara@regionprof.ru
УФА	(347) 292-5216	ufa@regionprof.ru
ЧЕЛЯБИНСК	(351) 239-9360	chelyabinsk@regionprof.ru

Реклама