Применение программы CodeVisionAVR для управления буквенно-цифровыми дисплеями в Proteus 8.11

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассмотрено применение функций программы CodeVisionAVR и генератора кода CodeWizardAVR для формирования анимации и отображения текста на экране буквенно-цифрового дисплея, работающего под управлением микроконтроллера AVR. Приведён пример моделирования схемы в Proteus с использованием микроконтроллера ATmega16 и датчика касания, пробников напряжения и логических уровней 0 и 1, буквенно-цифрового дисплея в 4- и 8-разрядных режимах работы. Проведён контроль входных/выходных сигналов, присутствующих на выводах микроконтроллера.

Введение

Программа Proteus позволяет автоматизировать все стадии проектирования электронных устройств, включая подготовку принципиальных схем, моделирование процессов, происходящих в электронных цепях, компоновку и трассировку печатных плат, редактирование и расширение библиотек компонентов. В Proteus реализовано большое количество функций для профессионального проектирования микроэлектронных устройств, ориентированных на самые современные средства моделирования. Одной из таких функций является имитация работы микроконтроллеров. Система, в которой используется микроконтроллер, может не только чем-то управлять, но и что-то отображать. Чаще всего в качестве узла отображения в микроэлектронной системе используют дисплеи, среди которых буквенно-цифровые (предназначенные для отображения информации в виде букв, цифр, различных знаков). Единичные элементы отображения таких индикаторов сгруппированы по строкам и столбцам. В Proteus буквенно-цифровые дисплеи находятся в разделе Alphanumeric LCDs библиотеки Optoelectronics и представлены следующими микросхемами: LM016L (16×2), LM017L (32×2), LM018L (40×2), LM020L (16×1), LM032L (20×2), LM041L (16×4), LM044L (20×4), MDLS40466 (40×4). В скобках указано разрешение дисплея. Например, 32×2 – 32 символа × 2 строки.

Для практического применения дисплея требуется его взаимодействие с внешним источником данных, генератором кодовых комбинаций символов, схемой управления, в качестве которых можно применить микроконтроллер. Каких-либо стандартных правил сопряжения микроконтроллеров с дисплеями не существует, и в каждом конкретном случае сопряжение может выполняться по-разному.

При проектировании в Proteus схемы устройства вывода информации, работающего под управлением микроконтроллера AVR, написание программы инициализации и её компиляцию можно выполнить с помощью CodeVisionAVR (интегрированной среды разработки программного обеспечения для микроконтроллеров семейства AVR). CodeVisionAVR поддерживает все базовые конструкции языка С, которые используются при написании программ (алфавит, константы, идентификаторы, комментарии) и разрешены архитектурой AVR, с некоторыми добавленными характеристиками, реализующими преимущество специфики архитектуры AVR. Используя специальные директивы, в любом месте программы можно включить ассемблерный код, что позволяет напрямую обращаться к регистрам микроконтроллера.

Формирование программного кода в CodeVisionAVR выполняют при

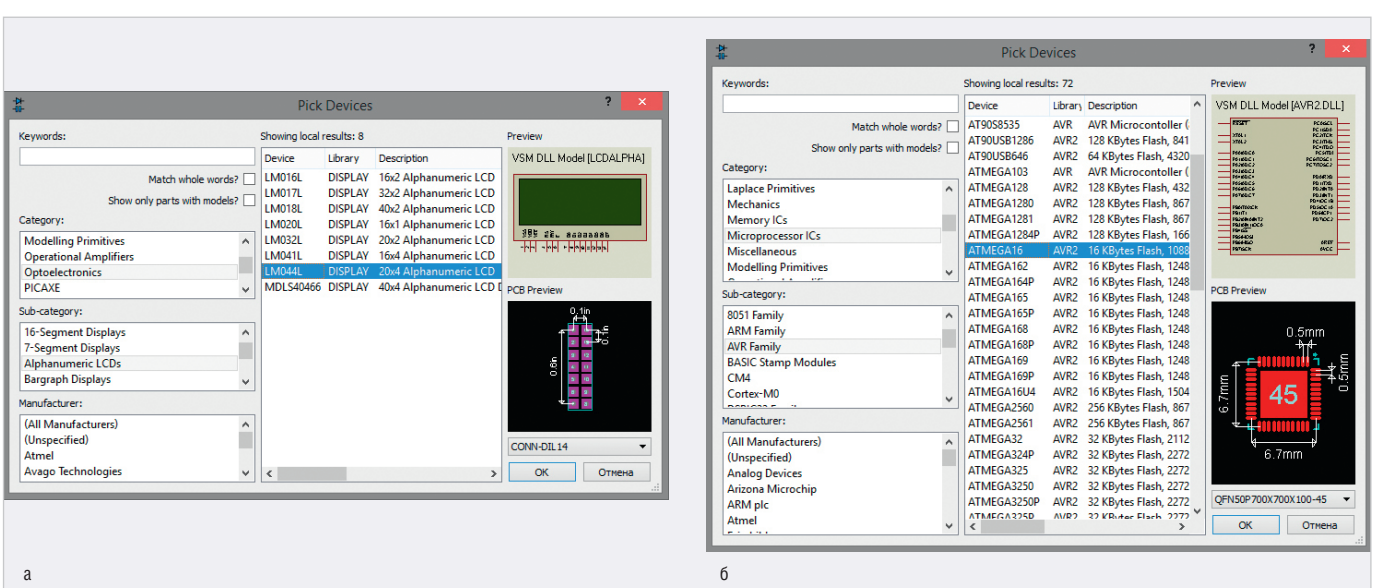


Рис. 1. Выбор микросхемы: (а) LM044L из раздела Alphanumeric LCDs библиотеки Optoelectronics, (б) ATmega16 из раздела AVR Family библиотеки Microprocessor ICs

помощи автоматического генератора CodeWizardAVR или вручную с нуля, используя синтаксис языка программирования С и функции стандартных библиотек программы для работы с буквенно-цифровыми LCD-модулями, шиной I^2C, EEPROM DS2430 и DS2433, температурными датчиками (LM75 и DS1820/DS18S20), часами реального времени (PCF8563, PCF8583, DS1302, DS1307), протоколом 1-Wire, SPI, USART, а также функции формирования задержек, управления питанием и преобразования кода Грея. Удобство применения генератора состоит в быстром получении кода выполнения функций инициализации микроконтроллера и его портов ввода/вывода, внешних прерываний, таймеров/счётчиков, сторожевого таймера, аналогового компаратора, интерфейсов USART и SPI, шин 1-Wire и I^2C, буквенно-цифровых и графических дисплеев, установки доступа к внешней памяти и др. Однако в процессе работы мастера формируется достаточно объёмный код, который впоследствии приходится редактировать.

Когда код пишется вручную, для управления буквенно-цифровым дисплеем применяют функции библиотеки для работы с буквенно-цифровыми модулями, функции ввода/вывода информации на экран дисплея, строковые функции языка С.

В статье рассмотрен вывод текстовых данных на экран дисплея LM044L (в 4- и 8-разрядном режиме), работающего под управлением программы инициализации микроконтроллера ATmega16, написанной в CodeVisionAVR на языке С с использованием системы команд контроллера HD44780. Применение функций управления буквенно-цифровым дисплеем программы CodeVisionAVR рассмотрено на примере схемы, где с помощью датчика касания и микроконтроллера ATmega16 проводится формирование управляющего сигнала, обработка полученных данных программой инициализации и вывод анимации в виде строки загрузки и информационных сообщений на экран буквенно-цифрового дисплея. Строка загрузки реализована последовательным заполнением знакомест первой строки экрана буквенно-цифрового дисплея (нумерация строк и столбцов начинается с нуля).

Проектирование схемы электрической принципиальной в Proteus

Буквенно-цифровые дисплеи – очень популярный способ вывода информации в электронных устройствах. Работу с буквенно-цифровыми дисплеями в Proteus рассмотрим на примере микросхемы LM044L с разрешением экрана 20 символов на 4 строки, для тестирования которой воспользуемся микроконтроллером ATmega16.

Если написание программного кода управления электронной системой предполагается выполнить в CodeVisionAVR, то проект схемы электрической принципиальной в схемном редакторе создают при помощи кнопки Schematic Capture верхней панели инструментов стартового окна Proteus. Нажатие кнопки открывает новую одноимённую вкладку, в рабочем поле которой и будет выполняться разработка схемы.

Создадим в Proteus новый проект и добавим в рабочее поле микросхему дисплея, для чего при помощи команды контекстного меню Place/Component/From Libraries схемного редактора откроем окно Pick Devices и выберем левой кнопкой мыши из раздела Alphanumeric LCDs библиотеки Optoelectronics микросхему LM044L (рис. 1а). Нажмём на кнопку OK (окно Pick Devices будет закрыто) и разместим микросхему в рабочей области проекта.

Микросхема LM044L работает под управлением контроллера HD44780, который принимает и обрабатывает команды управления и выводит соответствующие символы на дисплей. Микросхема имеет 14 контактов, назначение которых следующее:

- Vss – GND;
- Vdd – напряжение питания +5 В;
- Vee – напряжение контрастности от 0 до +5 В (настройка контрастности отображаемых на дисплее символов);
- RS – выбор регистра данных DR (RS – 1) или команд IR (RS – 0);
- RW – выбор операции чтения (RW – 1) или записи (RW – 0);
- E – линия синхронизации;
- D0...D7 – шина данных/команд.

Микросхема LM044L может работать в двух режимах:

- 8-разрядном (для обмена информацией используются выводы D0...D7);
- 4-разрядном (для обмена информацией используются выводы D4...D7).

Для тестирования работы дисплея будем использовать микроконтроллер, в качестве которого применим микросхему ATmega16 из раздела AVR Family библиотеки Microprocessor ICs (рис. 1б).

Для подключения микросхемы LM044L к схеме управления используется параллельная синхронная шина данных/команд (D0...D7), вывод выбора операции чтения/записи (RW), вывод выбора регистра данных/команд (RS) и вывод синхронизации (E). Подсоединим выводы модуля дисплея D0...D7 к выводам PD0...PD7 порта PD (для работы в 8-разрядном режиме), а выводы RS и E к выводам PC4 и PC7 порта PC микроконтроллера ATmega16 так, как показано на рис. 2а. Для работы в 4-разрядном режиме выводы модуля дисплея D4...D7 подсоединяют к выводам PD4...PD7 порта PD (рис. 2б). Вывод RW подключим к «земле», так как в нашей системе будет выполняться только запись информации в микросхему LM044L. Выводы Vss и Vdd подключим к «земле» и напряжению +5 В соответственно. На вывод Vee подаётся напряжение контрастности (от 0 до +5 В). На практике этот вывод подключается 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 (рис. 3). Панель открывают нажатием кнопки Terminals Mode на левой панели схемного редактора.

Приём информации микросхемой LM044L выполняется по 8-разрядной шине данных/команд в 8- и 4-разрядном режиме. Подача управляющих сигналов через подключённые к портам микроконтроллера ATmega16 линии выполняется программно. Выбор линий портов микроконтроллера для подключения к указанным выводам дисплея производится разработчиком произвольно.

В окне свойств буквенно-цифрового дисплея (окно открывают двойным щелчком левой кнопки мыши после его выделения на схеме) в поле Advanced Properties из выпадающего списка выбирают пункт Clock Frequency – тактовая частота (рис. 4а). Её значение должно совпадать с частотой работы микроконтроллера (в нашем примере 2 МГц), которую указывают в окне его свойств (рис. 4б). Для этого в поле Advanced Properties из выпадающего списка выбирают пункт Clock Frequency и определяют соответствующее ему значение (2 МГц). Также в окне свойств микроконтроллера установим следующие параметры:

- поле CKOPT (Oscillator Options) – (1) Unprogrammed;
- поле BOOTRST (Select Reset Vector) – (1) Unprogrammed;
- поле CKSEL Fuses – (0010) Int.RC 2MHz;

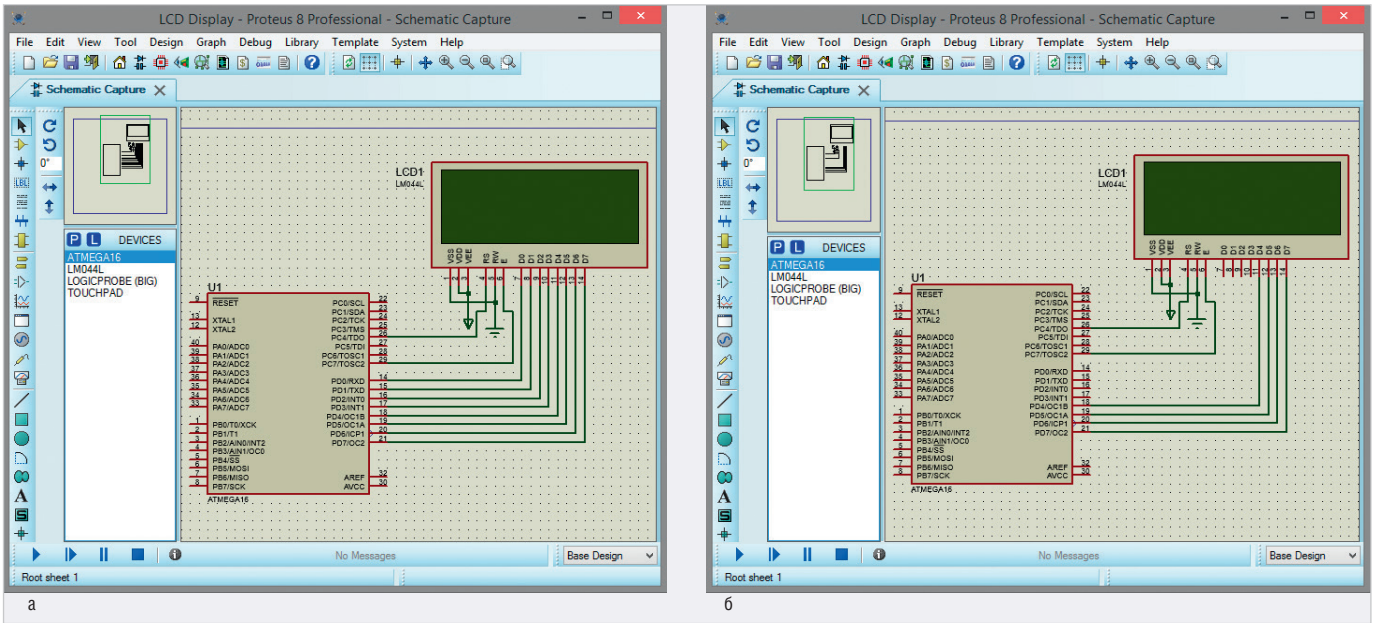


Рис. 2. Сопряжение микроконтроллера ATmega16 с буквенно-цифровым дисплеем LM044L, работающим в: (а) 8-разрядном и (б) 4-разрядном режиме

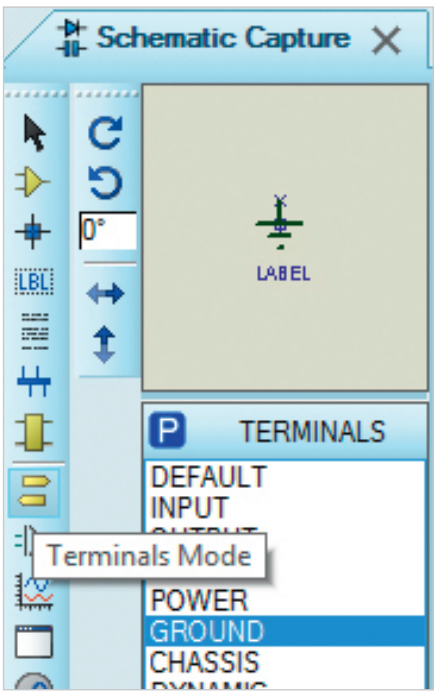


Рис. 3. Открытие при помощи кнопки Terminals Mode панели TERMINALS и выбор символа «земли»

Таблица 1. Система команд контроллера HD44780

Команда		Код								
		RS	D7	D6	D5	D4	D3	D2	D1	D0
Очистить дисплей и установить курсор в нулевую позицию		0	0	0	0	0	0	0	0	1
Возврат курсора в нулевую позицию		0	0	0	0	0	0	0	1	–
Выбор направления сдвига курсора при записи следующего символа (I/D – 1 сдвиг вправо, I/D – 0 сдвиг влево); Разрешение или запрет сдвига экрана (S – 1 сдвиг разрешён, S – 0 сдвиг запрещён)		0	0	0	0	0	0	1	I/D	S
Включить/выключить дисплей (D – 1 дисплей включён, D – 0 дисплей отключён); Включить/выключить отображение курсора на экране (C – 1 курсор отображается, C – 0 курсор не отображается); Включить/выключить мигание курсора (B – 1 курсор мигает, B – 0 курсор не мигает)		0	0	0	0	0	1	D	C	B
Сдвиг курсора/экрана (S/C – 1 сдвиг экрана, S/C – 0 сдвиг курсора; R/L – 1 сдвиг вправо, R/L – 0 сдвиг влево)		0	0	0	0	1	S/C	R/L	–	–
Выбор режима работы (DL – 1 8-разрядный, DL – 0 4-разрядный); Выбор количества используемых для работы строк экрана (N – 1 две строки, N – 0 одна строка); Выбор размера отображаемых на экране символов (F – 1 шрифт 5×7 пикселей, F – 0 шрифт 5×10 пикселей)		0	0	0	1	DL	N	F	–	–
Выбор адреса (ACG) ячейки памяти CGRAM		0	0	1	ACG	ACG	ACG	ACG	ACG	ACG
Выбор адреса (ADD) ячейки памяти DDRAM		0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD
Запись данных в выбранную ячейку памяти DDRAM или CGRAM	8-разрядный режим	1	8 бит	7 бит	6 бит	5 бит	4 бит	3 бит	2 бит	1 бит
	4-разрядный режим	1	8 бит	7 бит	6 бит	5 бит	–	–	–	–
			4 бит	3 бит	2 бит	1 бит	–	–	–	–

- поле Boot Loader Size – (00) 1024 words. Starts at 0x1C00;
- поле SUT Fuses – (01);
- поле Program File – путь к .hex (или .cof) файлу на диске компьютера.

Подача управляющих сигналов через подключённые к портам микроконтроллера ATmega16 линии выполняется программно в соответствии с табл. 1.

Необходимо отметить, что контроллер HD44780 содержит три вида памяти:

- DDRAM – оперативное запоминающее устройство, в котором хранятся коды символов, отображаемых на экране;
- CGROM – постоянное запоминающее устройство, которое содержит «битовое изображение» символов;
- CGRAM – оперативное запоминающее устройство, является частью CGROM, предназначено для хранения символов пользователя.

Последовательность действий, которые необходимо выполнить управляющей схеме при совершении операции записи по 8-разрядной шине, может быть следующей:

- установить значение линии RW=0 (запись в микросхему LM044L);
- установить значение линии RS=0 (приём команд);
- вывести значение байта команды 00001111 на линии шины D0...D7 (команда включения дисплея);
- вывести значение байта команды 00110100 на линии шины D0...D7 (установка разрядности шины);
- вывести значение байта команды 00000001 на линии шины D0...D7 (очистка дисплея и установка курсора в нулевую позицию);
- установить значение линии RS=1 (приём данных);

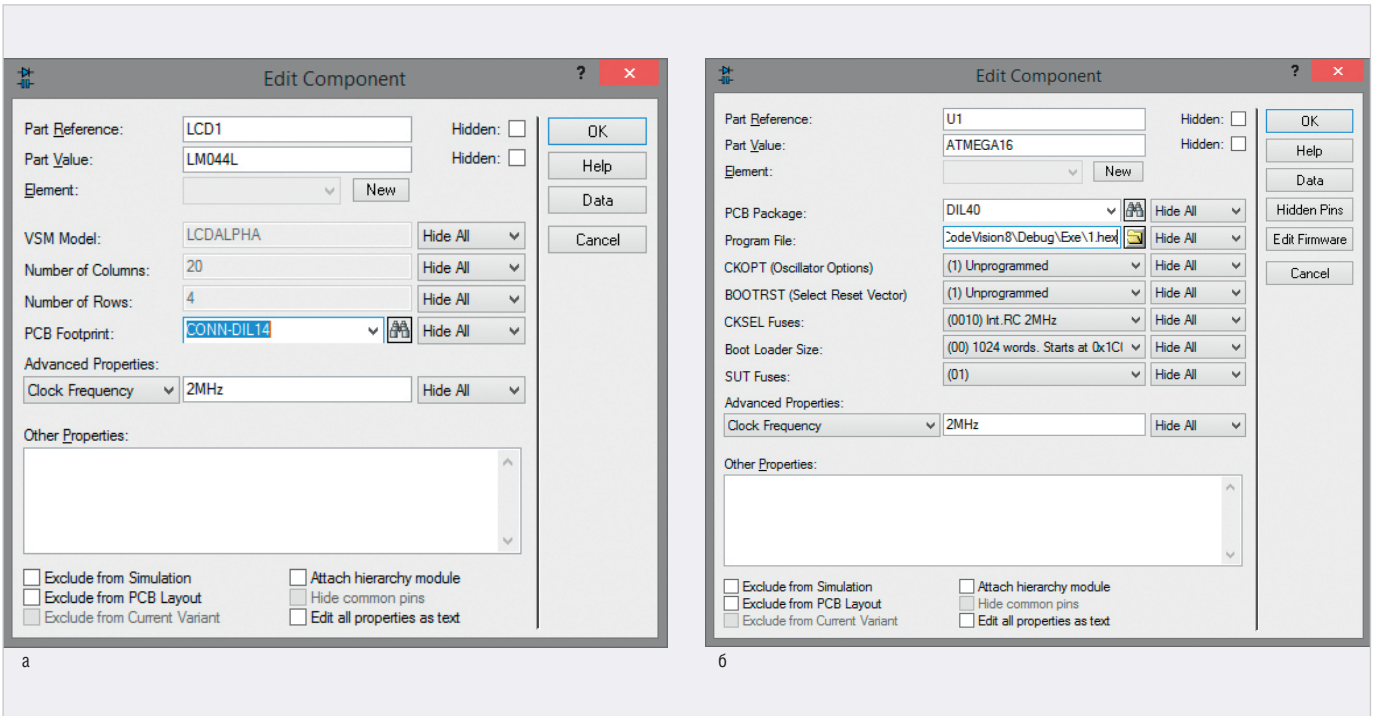


Рис. 4. Окно свойств: (а) буквенно-цифрового дисплея LM044L, (б) микроконтроллера ATmega16

- вывести значение байта данных на линии шины D0...D7.

Последовательность действий, которые необходимо выполнить управляющей схеме при совершении операции записи в 4-разрядном режиме, может быть следующей:

- установить значение линии RW=0 (запись в микросхему LM044L);
- установить значение линии RS=0 (приём команд);
- вывести значение старшего полубайта 0010 команды 00100000 на линии шины D4...D7 (установка разрядности шины);
- вывести значение старшего полубайта 0000, а затем младшего полубайта 1111 команды 00001111 на линии шины D4...D7 (команда включения дисплея);
- вывести значение старшего полубайта 0000, а затем младшего полубайта 0001 команды 00000001 на линии шины D4...D7 (очистка дисплея и установка курсора в нулевую позицию);
- установить значение линии RS=1 (приём данных);
- вывести значение старшего полубайта данных, а затем младшего полубайта данных на линии шины D4...D7.

При написании кода программы необходимо учитывать, что по умолчанию микросхема LM044L работает в 8-разрядном режиме, поэтому первую команду 00100000 (установка 4-разрядного режима) необходимо подать на шину данных/команд с учётом того, что информация будет передана за один такт. Так как в 4-разрядном режиме мы не имеем доступа к выводам D0...D3 микросхемы LM044L и управляющий бит находится в старшем полубайте (0010) команды (00100000), то младший полубайт (0000) на шину данных/команд D0...D7 выводить не нужно. После записи кода 0010 в регистр команд микросхема LM044L будет переведена в 4-разрядный режим работы, и дальнейшая передача информации будет выполняться через выводы D4...D7.

Необходимо учитывать, что большинство операций, выполняемых контроллером микросхемы LM044L, занимают значительное время, около 40 мкс, а время выполнения некоторых доходит до единиц миллисекунд. Поэтому в программе управления жидкокристаллическим модулем совершению любой операции должны предшествовать команды задержки, которые можно организовать с помощью функции delay_ms( ), где в скобках указывается время задержки в мс. Также необходимо обеспечить формирование тактового сигнала на линии E микросхемы LM044L. Сделать это можно программно, посредством чередования подачи значений нуля и единицы.

На языке программирования C эти действия можно реализовать следующим образом:

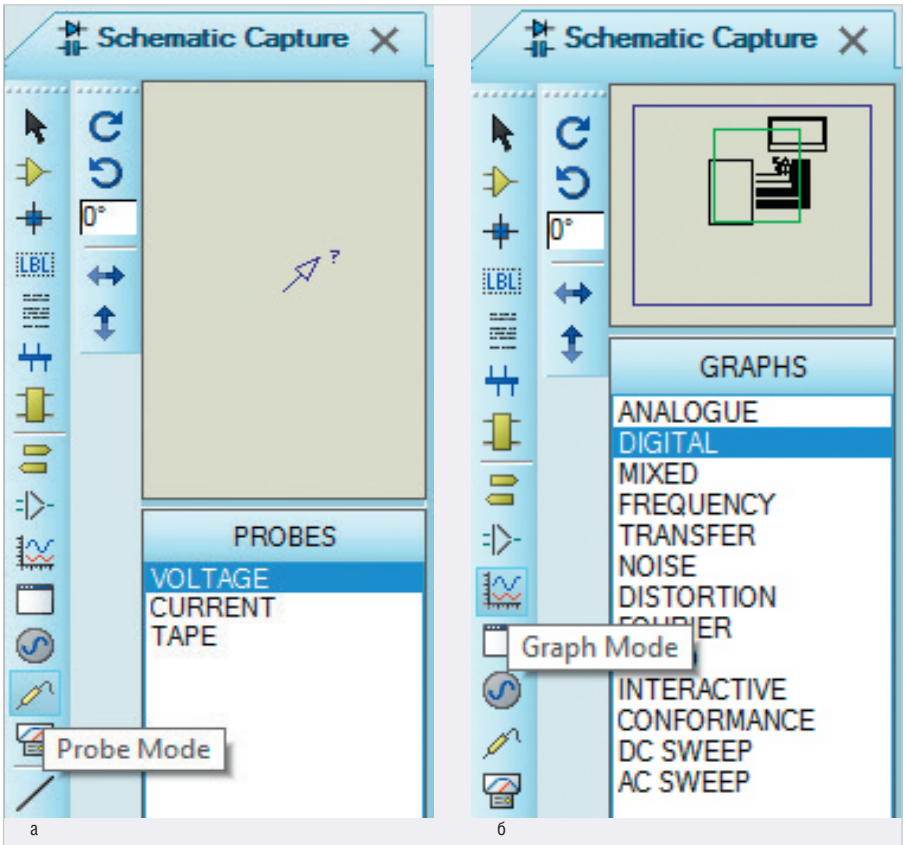
- для 8-разрядного режима работы:

```
PORTC &=~ (1<<4); // RS=0 (приём команд)
e ( ); PORTD=0b00001111; // включение дисплея
e ( ); PORTD=0b00110100; // установка 8-разрядной шины
e ( ); PORTD=0b00000001; // очистка дисплея и установка курсора в нулевую позицию
while(1) {
e ( ); PORTC |= 1<<4; } // RS=1 (приём данных)
```

- для 4-разрядного режима работы:

```
PORTC &=~ (1<<4); // RS=0 (приём команд)
e ( ); PORTD=0b00100000; //установка 4-разрядной шины
// команда включения дисплея 00001111
e ( ); PORTD=0b00000000; //старший полубайт 0000 кода 00001111
e ( ); PORTD=0b11110000; //младший полубайт 1111 кода 00001111
//команда очистки дисплея и установки курсора в нулевую позицию 00000001
e ( ); PORTD=0b00000000; //старший полубайт 0000 кода 00000001
e ( ); PORTD=0b00010000; //младший полубайт 0001 кода 00000001
while(1) {
e ( ); PORTC |= 1<<4; } // RS=1 (приём данных)
```

В данном примере для передачи команд/данных используется порт PD микроконтроллера. Для передачи управляющих сигналов задействован ещё один порт – PC. Функция e ( ); обеспечивает формирование тактового сигнала на линии E дисплея.



а б

Рис. 5. Выбор: (а) пробника VOLTAGE на панели PROBES, (б) графика DIGITAL на панели GRAPHS

Контроль входных/выходных сигналов, присутствующих на выводах микроконтроллера, в нашем примере организован при помощи пробников напряжения VOLTAGE. Пробники разместим с помощью мыши до запуска процесса симуляции в тех точках схемы, за которыми мы хотим наблюдать. Выбор пробников осуществляется на панели PROBES (рис. 5а), она открывается нажатием кнопки Probe Mode на левой панели инструментов схемного редактора.

Для отображения данных, снятых пробниками со схемы, нужно добавить пробники на график, выделить его левой кнопкой мыши и выполнить моделирование командой контекстного меню Simulate Graph, которое вызывают щелчком правой кнопкой мыши. Необходимо отметить, что измерительные пробники не имеют собственной лицевой панели, как другие виртуальные приборы. А настройка их параметров осуществляется в окне свойств до запуска симуляции схемы.

Для анализа цифровых сигналов подойдёт график DIGITAL, добавление которого в рабочую область проекта выполняют выбором его названия из списка на панели GRAPHS (рис. 5б). Панель открывают нажатием кнопки Graph Mode на левой панели инструментов схемного редактора. Далее график, который фактически является окном отображения результатов анализа, размещают с помощью мыши в необходимом месте рабочего поля программы. Для этого помещают указатель мыши в окне редактора в точке, где должен находиться верхний левый угол графика. Нажимают левую кнопку мыши и растягивают прямоугольник до того размера, который необходим для отображения результатов анализа, а затем отпускают кнопку мыши.

Для подключения измерительного пробника к схеме необходимо выбрать его название на панели PROBES, подвести курсор к месту размещения пробника и щёлкнуть левой кнопкой мыши по проводнику. В каждой схеме может использоваться много пробников, в том числе и копии одного и того же прибора. Каждая копия прибора настраивается и соединяется отдельно. Неподключённые пробники имеют по умолчанию название «?». Когда пробник присоединён к цепи, ему автоматически присваивается имя цепи, а если цепь не имеет имени, то пробник получает в качестве имени позиционное обозначение компонента или имя вывода, после которого он подключён. Также разработчик может самостоятельно присвоить пробнику имя.

Каждый график может содержать несколько диаграмм. Каждая диаграмма отображает данные, ассоциированные с одним пробником. Таким образом, для исследования работы собранной в нашем примере схемы в рабочее поле проекта необходимо добавить пробники напряжения для контроля сигналов на линиях шины данных дисплея, а также пробники для контроля тактового сигнала на линии E дисплея и сигнала выбора регистра данных или команд дисплея.

По умолчанию названия диаграмм на графике соответствуют названиям объектов, данные которых они отображают (рис. 6). Если нужно, названия диаграмм можно изменить. Для отображения на графике данных, зафиксированных пробниками, необходимо добавить на него эти объекты. Причём график должен быть размещён в рабочем поле проекта. Для добавления пробника на график нужно левой кнопкой мыши выделить его пиктограмму на схеме и перетащить её мышью в окно графика. В результате на графике отобразится название добавленного пробника.

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 Schematic Capture.

Создание программного кода в CodeVisionAVR

Командой основного меню File/New/Project создадим новый проект в CodeVisionAVR. В процессе создания:

- откажемся от применения генератора CodeWizardAVR для формирования программного кода, для чего в окне Confirm нажмём на кнопку No (рис. 7);
- в открывшемся окне Create New Project выберем директорию размещения нового проекта, имя проекта (поле «Имя файла»), его тип (поле «Тип файла») и нажмём кнопку «Сохранить» (рис. 8);
- в открывшемся окне New Project в поле Name выполним выбор микроконтроллера ATmega16 (рис. 9), под

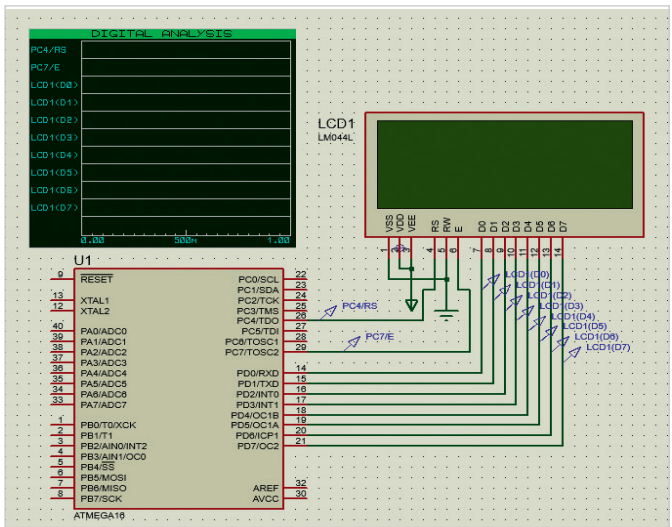


Рис. 6. Подключение пробников напряжения к схеме управления буквенно-цифровым дисплеем в 8-разрядном режиме работы и их размещение на графике

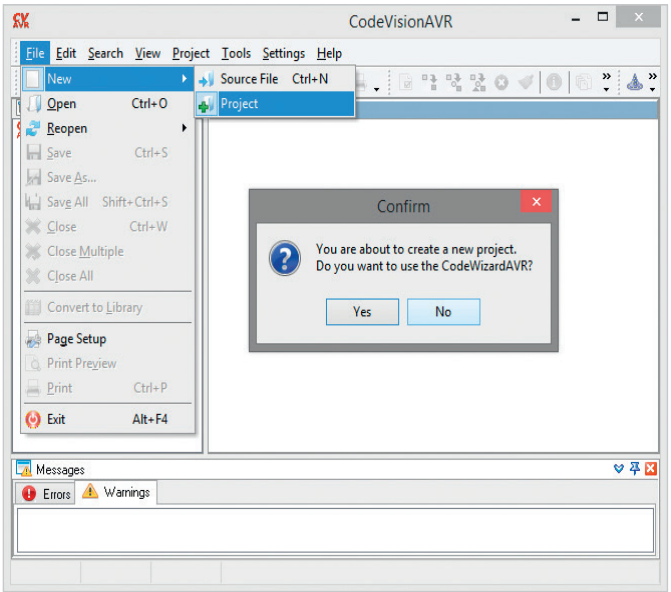


Рис. 7. Создание нового проекта в CodeVisionAVR

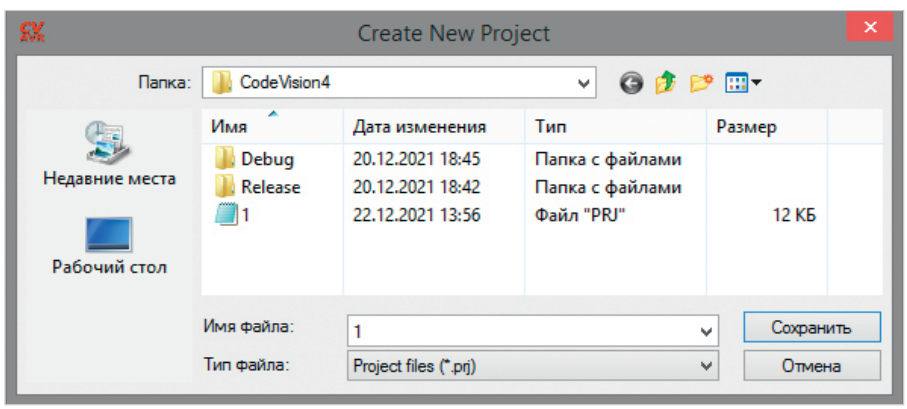


Рис. 8. Окно выбора директории размещения нового проекта Create New Project

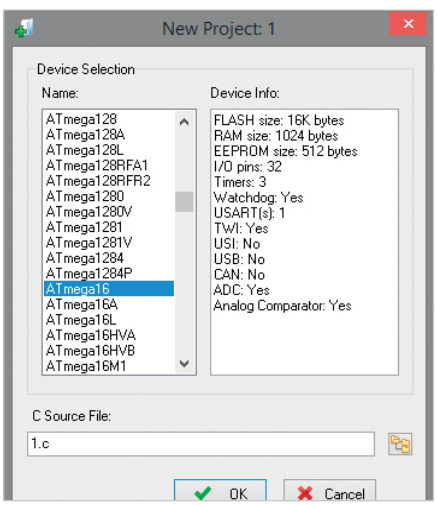


Рис. 9. Выбор микроконтроллера ATmega16 в окне New Project

управлением которого работает собранная схема (его описание отобразится в поле Device Info) и нажмём кнопку OK;

- в открывшемся окне настройки параметров проекта CodeVisionAVR (Configure Project) перейдём на вкладку C Compiler, на которой выберем закладку Code Genereration (рис. 10), где укажем:
  - размер стека данных в байтах (поле Data Stack Size) – для компиляции кода в нашем примере значения 256 будет достаточно;
  - размер кучи (поле Heap Size) и внутренней (поле Internal RAM Size) оперативной памяти – 0 и 1024 байт соответственно;
  - тактовую частоту микроконтроллера (поле Clock) – 2 МГц;
  - модель памяти (поле Memory Model) – Small.

Другие параметры оставим без изменений и нажмём на кнопку OK. В результате будет создан новый проект CodeVisionAVR, в окне кода которого и будет вестись дальнейшее написание программы.

Используя систему команд контроллера HD44780, напишем на языке C программу для микроконтроллера ATmega16, которая в качестве примера будет непрерывно выводить на экран дисплея, работающего в 8-разрядном режиме, строку «CodeVisionAVR». Для этого, исходя из таблицы соответствия символов английского алфавита и двоичного кода (табл. 2), представим символы строки «CodeVisionAVR» в двоичном коде:

```
C 01000011
o 01101111
d 01100100
e 01100101
V 01010110
i 01101001
s 01110011
i 01101001
o 01101111
n 01101110
A 01000001
V 01010110
R 01010010
```

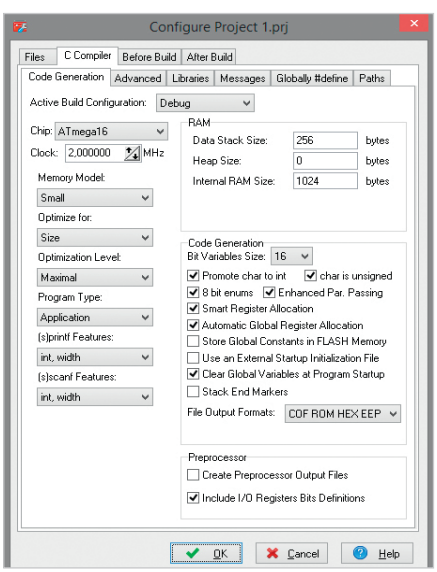


Рис. 10. Закладка Code Generation вкладки C Compiler окна настройки параметров проекта CodeVisionAVR, в котором используется микроконтроллер ATmega16

Таблица 2. Таблица соответствия символов английского алфавита и двоичного кода

Символ	Двоичный код	Символ	Двоичный код
A	01000001	a	01100001
B	01000010	b	01100010
C	01000011	c	01100011
D	01000100	d	01100100
E	01000101	e	01100101
F	01000110	f	01100110
G	01000111	g	01100111
H	01001000	h	01101000
I	01001001	i	01101001
J	01001010	j	01101010
K	01001011	k	01101011
L	01001100	l	01101100
M	01001101	m	01101101
N	01001110	n	01101110
O	01001111	o	01101111
P	01010000	p	01110000
Q	01010001	q	01110001
R	01010010	r	01110010
S	01010011	s	01110011
T	01010100	t	01110100
U	01010101	u	01110101
V	01010110	v	01110110
W	01010111	w	01110111
X	01011000	x	01111000
Y	01011001	y	01111001
Z	01011010	z	01111010

Программа вывода строки «CodeVisionAVR» на экран дисплея, работающего в 8-разрядном режиме:

```
  #include <io.h>
  #include <mega16.h> // подключение заголовочных файлов
  #include <stdio.h> // в которых содержатся
  #include <delay.h> // прототипы функций
  #include <stdlib.h>

  void e ( ) { // функция формирования тактового сигнала на линии E дисплея
  PORTC |= 1<<7; // устанавливаем 1 на выводе PC7 микроконтроллера
  delay_ms(40); // задержка 40 мс
  PORTC &=~ (1<<7); // устанавливаем 0 на выводе PC7 микроконтроллера
  delay_ms(40);
  }

  void main() { // основная функция программы
  DDRD=DDRC=0xff;  // инициализация портов
  PORTD=0x00; // порт PD работает на вывод данных
  PORTC=0x00; // порт PC работает на вывод данных
```

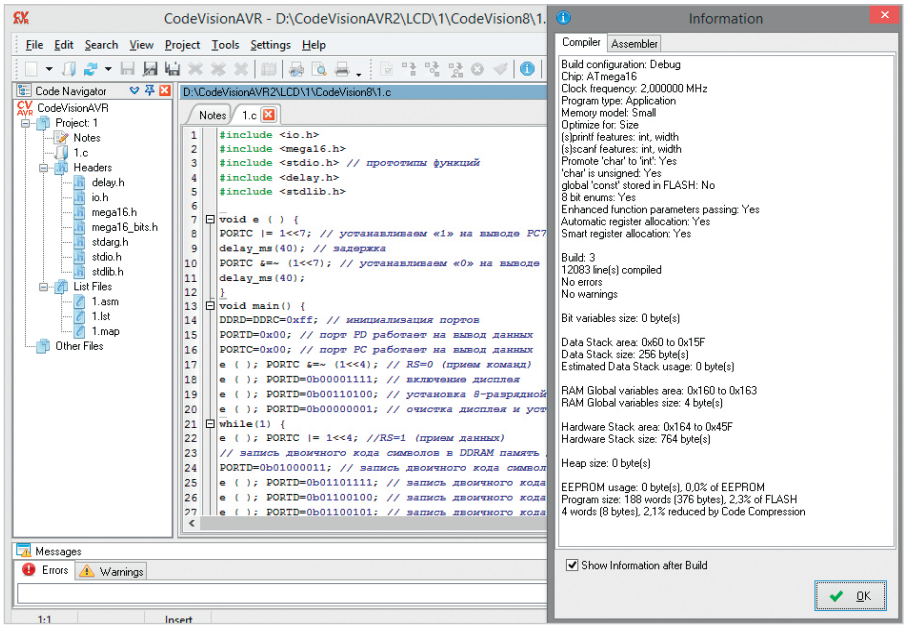


Рис. 11. Программа вывода строки на экран буквенно-цифрового дисплея в окне кода CodeVisionAVR и результат её компиляции

```
  e ( ); PORTC &=~ (1<<4); // RS=0 (приём команд)
  e ( ); PORTD=0b00001111; // включение дисплея
  e ( ); PORTD=0b00110100; // установка 8-разрядной шины
  e ( ); PORTD=0b00000001; // очистка дисплея и установка курсора в нулевую позицию
  while(1) {
  e ( ); PORTC |= 1<<4; // RS=1 (приём данных)
  // запись двоичного кода символов в DDRAM память для их отображения на дисплее
  PORTD=0b01000011; // запись двоичного кода символа C
  e ( ); PORTD=0b01101111; // запись двоичного кода символа o
  e ( ); PORTD=0b01100100; // запись двоичного кода символа d
  e ( ); PORTD=0b01100101; // запись двоичного кода символа e
  e ( ); PORTD=0b01010110; // запись двоичного кода символа V
  e ( ); PORTD=0b01101001; // запись двоичного кода символа i
  e ( ); PORTD=0b01110011; // запись двоичного кода символа s
  e ( ); PORTD=0b01101001; // запись двоичного кода символа i
  e ( ); PORTD=0b01101111; // запись двоичного кода символа o
  e ( ); PORTD=0b01101110; // запись двоичного кода символа n
  e ( ); PORTD=0b01000001; // запись двоичного кода символа A
  e ( ); PORTD=0b01010110; // запись двоичного кода символа V
  e ( ); PORTD=0b01010010; // запись двоичного кода символа R
  }}
```

Введём текст программы в окне кода CodeVisionAVR и запустим командой основного меню Project/Build All компиляцию, по окончании которой выдаётся отчёт о наличии ошибок в коде программы (рис. 11). При этом в группе Headers на панели Code Navigator отобразится список заголовочных файлов с расширением *.h, функции которых применяются в программе. Если ошибки не обнаружены, на диске компьютера будут созданы файлы .hex и .cof для записи в микроконтроллер.

Перейдём в редактор Schematic Capture программы Proteus, откроем окно свойств микроконтроллера и в поле Program File укажем путь к файлу прошивки на диске компьютера, полученному в результате компиляции программного кода (при условии отсутствия в коде ошибок). Кнопкой Run the simulation, расположенной в левом нижнем углу окна редактора, или командой основного меню Debug/Run Simulation запустим моделирование собранной схемы, результат которого представлен на рис. 12а.

Проанализируем работу демонстрационной схемы, представленной на рисунке 12а. В представленном примере даны указания микроконтроллеру через порт PD отправить контроллеру микросхемы LM044L кодовые комбинации команд (если на линии PC4 ноль) или данные (если на линии PC4 еди-

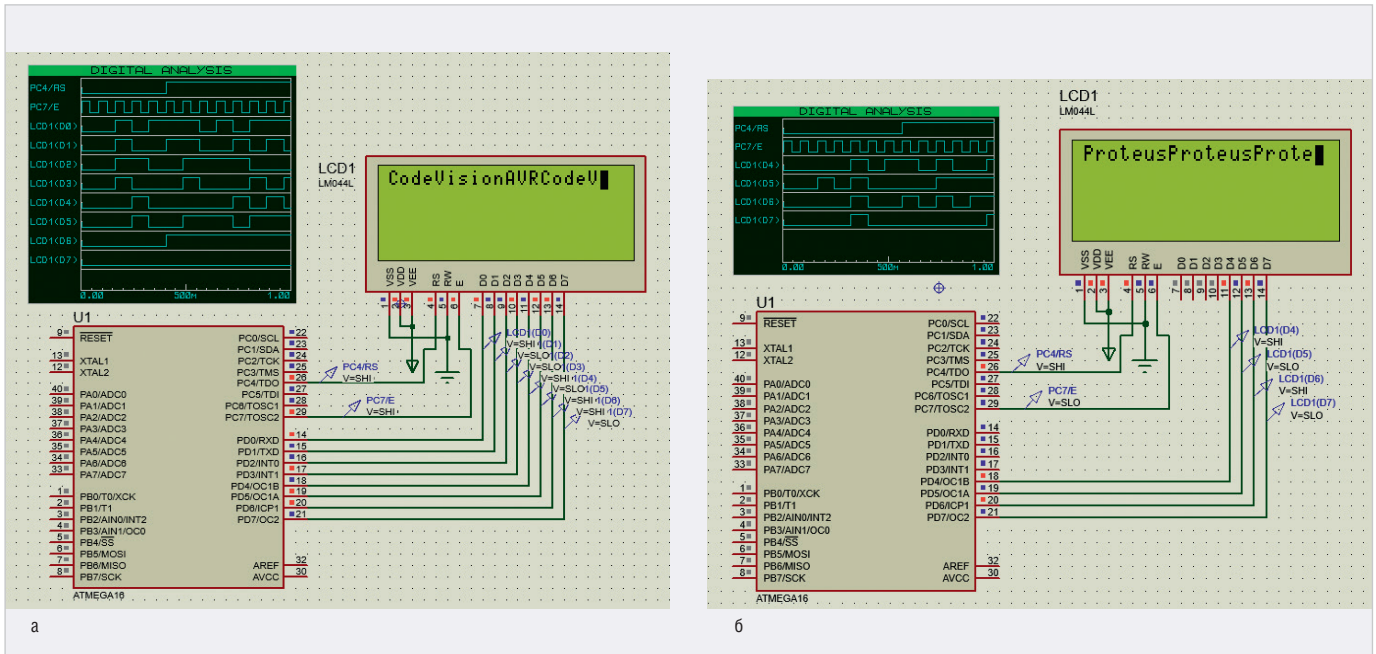


Рис. 12. Результат работы схемы управления буквенно-цифровым дисплеем: (а) в 8-разрядном режиме, (б) в 4-разрядном режиме и диаграммы сигналов, присутствующих на выводах микроконтроллера

ница). Для приёма команд/данных в микросхеме LM044L используются линии D0...D7. Управляющий сигнал с линии PC4 порта PC поступает на вывод RS микросхемы LM044L и подаётся программно. Вывод PC7 микроконтроллера подключён к выводу E микросхемы LM044L и используется для подачи тактовых импульсов.

После запуска симуляции схемы программа инициализации микроконтроллера выводит на линию PC4 логический ноль, который поступает на вывод RS микросхемы LM044L, в результате чего шина D0...D7 переходит в режим приёма следующих команд: включение дисплея, установка 8-разрядной шины, очистка дисплея и установка курсора в нулевую позицию. Далее программа инициализации микроконтроллера выводит на линию PC4 логическую единицу, что переводит шину D0...D7 микросхемы LM044L в режим приёма данных, запись которых выполняется побайтно в цикле. При этом на вывод E непрерывно подаётся тактовый сигнал, по заднему фронту которого микросхема LM044L считывает информацию (команды/данные). Таким образом, на экран дисплея посимвольно выводится строка, двоичные коды символов которой были поданы на шину D0...D7.

Программа вывода строки «Proteus» на экран дисплея, работающего в 4-разрядном режиме:

```
#include <io.h> // подключение заголовочных файлов
#include <mega16.h> // в которых содержатся
#include <stdio.h> // прототипы функций
#include <delay.h>
#include <stdlib.h>
void e ( ) { // функция формирования тактового сигнала на линии E дисплея
PORTC |= 1<<7; //устанавливаем 1 на выводе PC7 микроконтроллера
delay_ms(40); //задержка 40 мс
PORTC &=~ (1<<7); //устанавливаем 0 на выводе PC7 микроконтроллера
delay_ms(40); }

void main() {
DDRD=DDRC=0xff; // инициализация портов
PORTD=0x00; // порт PD работает на вывод данных
PORTC=0x00; // порт PC работает на вывод данных
e ( ); PORTC &=~ (1<<4); // RS=0 (приём команд)
e ( ); PORTD=0b00100000; // установка 4-разрядной шины
// команда включения дисплея 00001111
e ( ); PORTD=0b00000000; //старший полубайт 0000 кода 00001111
e ( ); PORTD=0b11110000; //младший полубайт 1111 кода 00001111
//команда очистки дисплея и установки курсора в нулевую позицию 00000001
e ( ); PORTD=0b00000000; //старший полубайт 0000 кода 00000001
e ( ); PORTD=0b00010000; //младший полубайт 0001 кода 00000001
while(1) {
e ( ); PORTC |= 1<<4; //RS=1 (приём данных)
//запись двоичного кода символа P (01010000) в DDRAM память
//для его отображения на дисплее
PORTD=0b01010000; //старший полубайт 0101 кода 01010000
e ( ); PORTD=0b00000000; //младший полубайт 0000 кода 01010000
//запись двоичного кода символа r 01110010
e ( ); PORTD=0b01110000; //старший полубайт 0111 кода 01110010
e ( ); PORTD=0b00100000; //младший полубайт 0010 кода 01110010
//запись двоичного кода символа o 01101111
e ( ); PORTD=0b01100000; //старший полубайт 0110 кода 01101111
e ( ); PORTD=0b11110000; //младший полубайт 1111 кода 01101111
//запись двоичного кода символа t 01110100
e ( ); PORTD=0b01110000; //старший полубайт 0111 кода 01110100
e ( ); PORTD=0b01000000; //младший полубайт 0100 кода 01110100
//запись двоичного кода символа e 01100101
e ( ); PORTD=0b01100000; //старший полубайт 0110 кода 01100101
e ( ); PORTD=0b01010000; //младший полубайт 0101 кода 01100101
//запись двоичного кода символа u 01110101
```

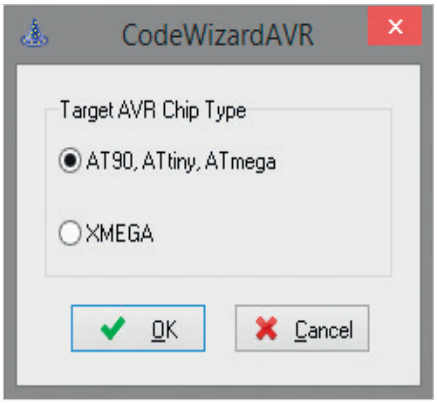


Рис. 13. Выбор типа микроконтроллера AVR

```
  e ( ); PORTD=0b01110000; //старший полубайт 0111 кода 01110101
  e ( ); PORTD=0b01010000; //младший полубайт 0101 кода 01110101
  //запись двоичного кода символа s 01110011
  e ( ); PORTD=0b01110000; //старший полубайт 0111 кода 01110011
  e ( ); PORTD=0b00110000; //младший полубайт 0011 кода 01110011
  }}
```

После запуска симуляции схемы программа инициализации микроконтроллера выводит на линию PC4 логический ноль, который поступает на вывод RS микросхемы LM044L. В результате шина D0...D7 переходит в режим приёма команд, первая из которых переводит микросхему LM044L в 4-разрядный режим работы. После этого через линии D4...D7 выполняется приём команд включения дисплея, очистки дисплея и установки курсора в нулевую позицию. Запись байта команды в регистр команд выполняется в два этапа: сначала на выводы D4...D7 подаётся старший полубайт команды, затем младший. Далее программа инициализации микроконтроллера выводит на линию PC4 логическую единицу, что переводит линии D4...D7 микросхемы LM044L в режим приёма данных, запись которых выполняется в цикле следующим образом: сначала на выводы D4...D7 подаётся старший полубайт данных, затем младший.

При этом на вывод E непрерывно подаётся тактовый сигнал, по заднему фронту которого микросхема LM044L считывает информацию (команды/данные). Таким образом, на экран дисплея посимвольно выводится строка, двоичные коды символов которой были поданы на шину D4...D7 (рис. 12б).

Символы строки «Proteus» в двоичном коде:

```
P 01010000
r 01110010
o 01101111
t 01110100
e 01100101
u 01110101
s 01110011
```

В CodeVisionAVR есть инструменты, с помощью которых написание программного кода значительно упрощается, а многие операции могут быть автоматизированы. Например, с помощью функции lcd_puts(char *str) библиотеки alcd.h можно заменить представленный выше фрагмент кода вывода на экран буквенно-цифрового дисплея строки Proteus одной командой lcd_puts(«Proteus»).

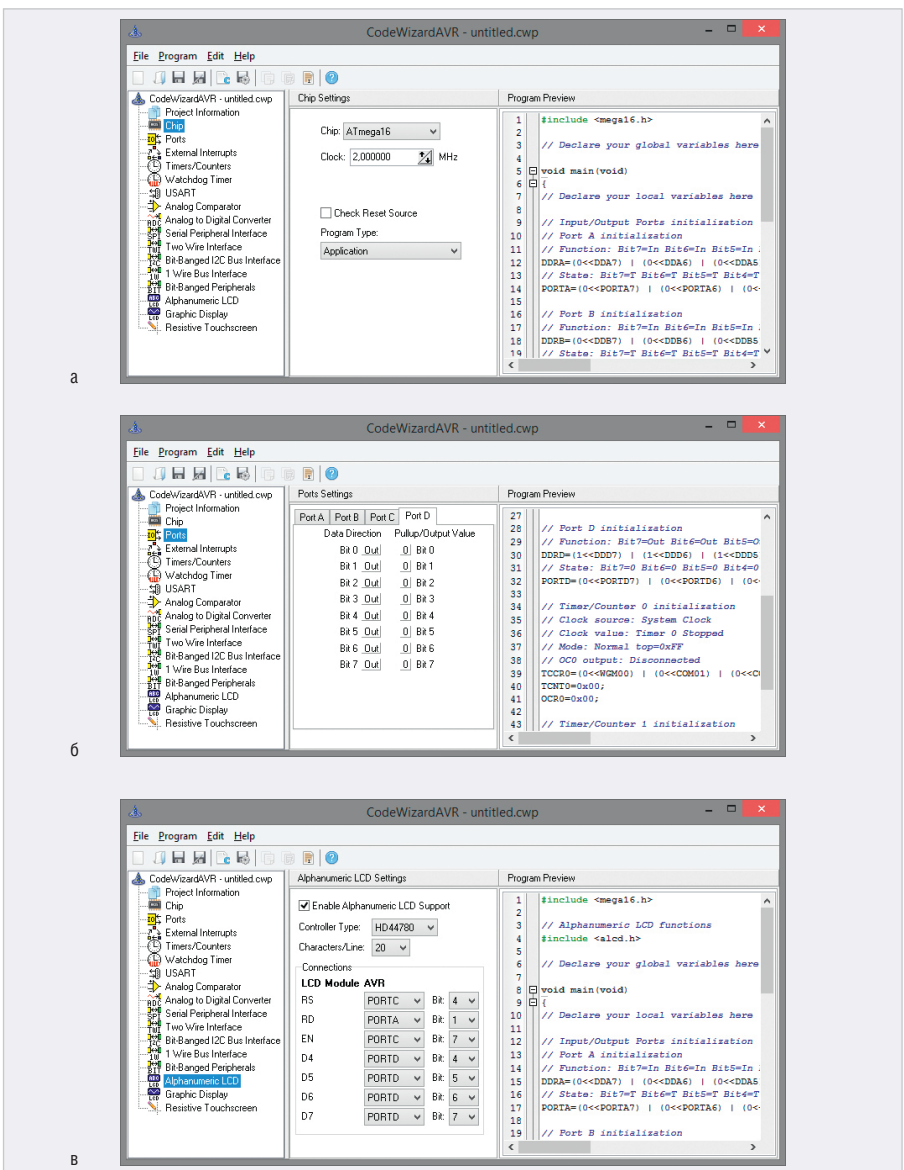


Рис. 14. Настройка в окне CodeWizardAVR параметров: (а) микроконтроллера, (б) портов ввода/вывода микроконтроллера, (в) буквенно-цифрового дисплея и подключения его выводов к микроконтроллеру

Применение CodeWizardAVR для формирования кода управления LCD-дисплеем

Воспользуемся для формирования программного кода управления буквенно-цифровым дисплеем генератором кода CodeWizardAVR, окно которого открывается в процессе создания командой основного меню File/New/Project нового проекта в CodeVisionAVR после нажатия кнопки Yes в окне Confirm (рис. 7). Перед запуском генератора будет предложено выбрать тип микроконтроллера (AT90, ATtiny, ATmega или XMEGA), что выполняют установкой переключателя в соответствующую позицию (рис. 13).

В CodeWizardAVR задают параметры микроконтроллера, его внутренних ресурсов и используемых в схеме периферийных устройств. В нашем примере это тип и частота работы микро-

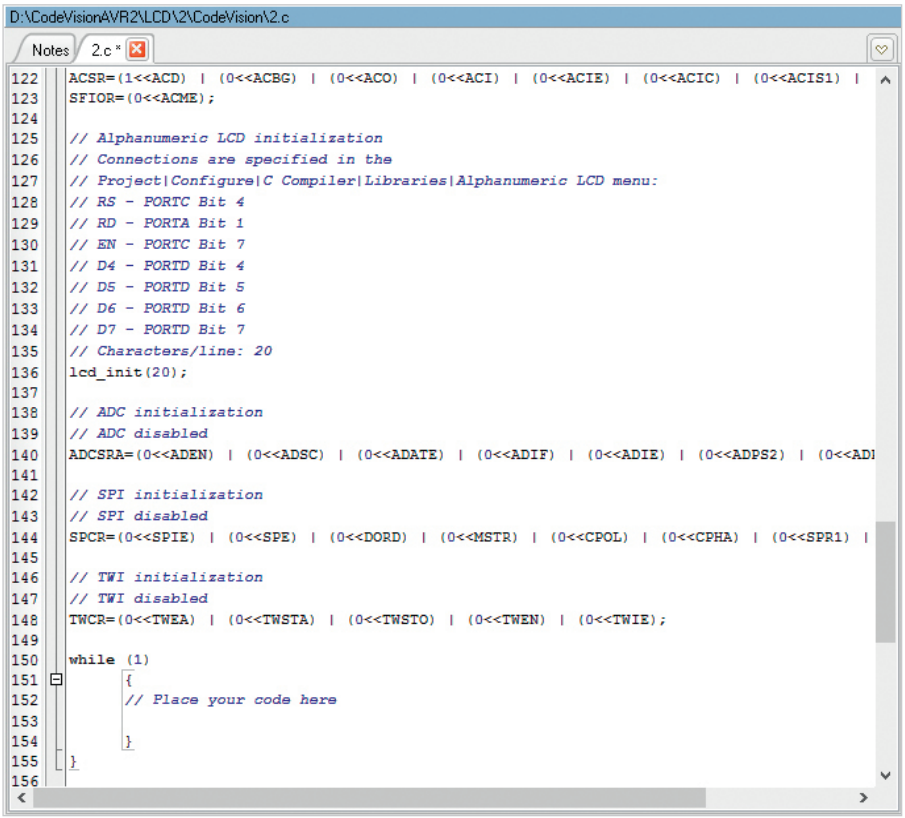


Рис. 15. Фрагмент программы, полученной с помощью генератора CodeWizardAVR, в окне кода CodeVisionAVR

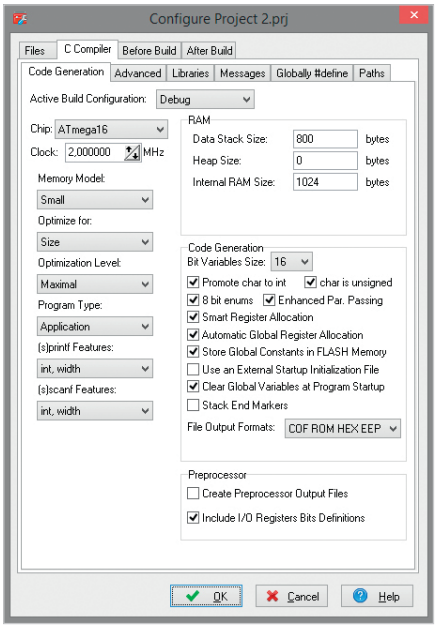


Рис. 16. Увеличение объёма стека данных до 800 байт в поле Data Stack Size закладки Code Generation окна Configure Project

контроллера (вкладка Chip Settings – рис. 14а), опции портов ввода/вывода микроконтроллера (вкладка Ports Settings – рис. 14б), буквенно-цифрового дисплея (вкладка Alphanumeric LCD Settings – рис. 14в). Важно, чтобы значение тактовой частоты микроконтроллера, указанное в поле Clock вкладки Chip Settings, совпадало со значением Clock Frequency в поле Advanced Properties его окна свойств в Proteus (в нашем примере это 2MHz).

На вкладке Ports Settings для каждого отдельного порта микроконтроллера отведена своя закладка, где в поле Direction щелчком левой кнопки мыши выбирают одно из значений битов порта: Out (линия порта работает на вывод данных), In (приём данных). В нашем примере (рис. 14б) для битов Bit 0...Bit 7 портов Port D, Port C укажем значение Out, для бита Bit 1 порта Port A – значение In.

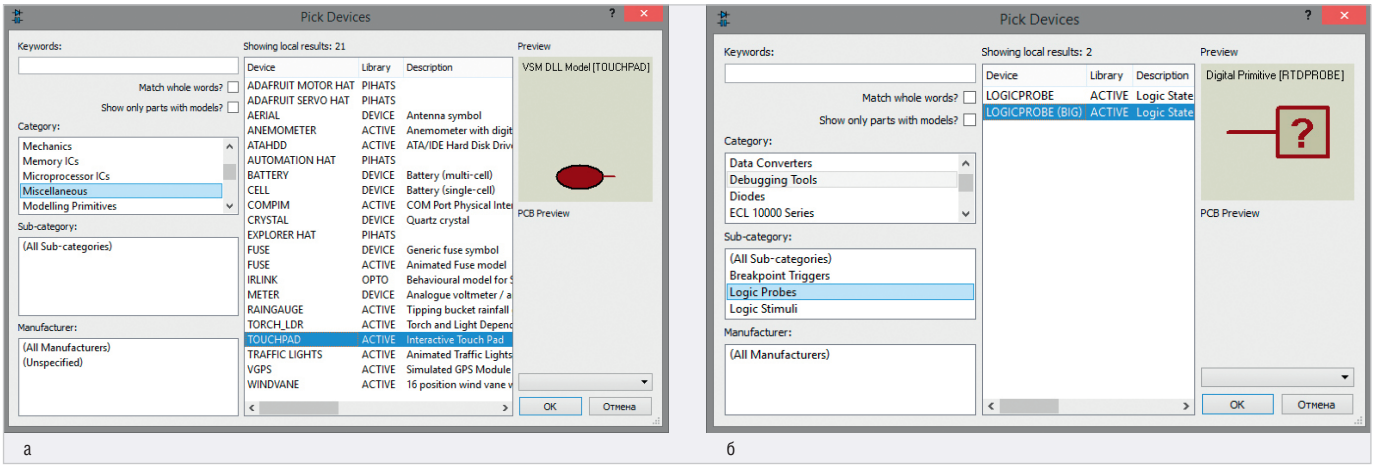
На вкладке Alphanumeric LCD Settings (рис. 14в) установкой флажка в чекбоксе задают разрешение поддержки буквенно-цифрового дисплея (поле Enable Alphanumeric LCD Support), тип контроллера (поле Controller Type, в нашем примере – HD44780) и количество символов в строке (поле Character/Line, в нашем примере – 20). В поле Connections настраивают параметры подключения микроконтроллера (порт и номер вывода) к микросхеме дисплея, работающего в 4-разрядном режиме. В нашем примере 4 и 7 биты порта PC микроконтроллера подключены к выводам RS и E дисплея, 4...7 биты порта PD микроконтроллера подключены к выводам D4...D7 дисплея. Вывод RD дисплея на схеме подключён к «земле», поэтому в поле Connections параметры его сопряжения с микроконтроллером можно не задавать. Если предполагается, что буквенно-цифровой дисплей будет работать в 8-разрядном режиме и написание кода программы управления будет вестись самостоятельно (так как стандартной библиотеки для этого режима в CodeVisionAVR нет), то поле Connections можно не заполнять.

Предварительный просмотр кода программы, который генерируют командой Program/Generate основного меню, выполняют в поле Program Preview. После настройки параметров и генерации кода командой Program/Generate, Save and Exit основного меню или пиктограммой Generate Program, Save and Exit верхней панели инструментов окно CodeWizardAVR автоматически будет закрыто. После выбора директории размещения нового проекта полученный код (рис. 15) отобразится в окне кода CodeVisionAVR, где и будет вестись дальнейшее написание программы. При этом автоматически сгенерированный код можно откорректировать на своё усмотрение.

Прежде чем внести изменения в полученный код в CodeVisionAVR, увеличим размер стека. Для этого командой Project/Configure основного меню откроем окно Configure Project, перейдём на вкладку C Compiler, где откроем закладку Code Generation и в поле Data Stack Size укажем размер стека в байтах – для компиляции кода в нашем примере значения 800 будет достаточно (рис. 16).

Применение функций библиотеки alcd.h для формирования кода управления LCD-дисплеем

Напишем на языке С программу для микроконтроллера ATmega16, которая в качестве примера будет выводить на экран дисплея строку WARNING! (если на линии PA1 микроконтроллера единица) и строку ___OKey___ (если на линии ноль). Для формирования управляющего сигнала в нашем примере в схему добавлен датчик касания TOUCHPAD, который находится в библиотеке компонентов Miscellaneous программы Proteus (рис. 17а). Датчик может работать в качестве сенсорной кнопки или сигнализировать о касании к устройству. Для работы с датчиком касания настроим линию PA1 на чтение данных, полученных от внешнего устройства, которые будут



а б

Рис. 17. Выбор в Proteus: (а) датчика касания TOUCHPAD из библиотеки Miscellaneous, (б) пробника логических уровней 0 и 1 LOGICPROBE (BIG) из раздела Logic Probes библиотеки Debugging Tools

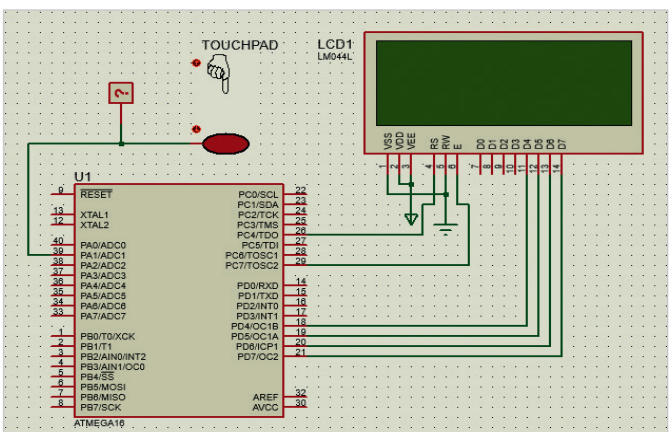


Рис. 18. Сопряжение буквенно-цифрового дисплея LM044L, микроконтроллера ATmega16 и датчика касания в схемном редакторе Proteus

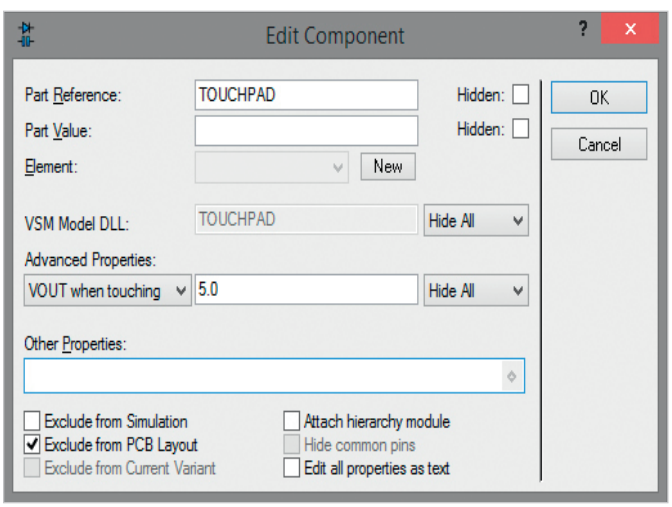


Рис. 19. Окно свойств датчика касания

записаны в переменную res. Нажатие датчика приводит к появлению на линии PA1 логической единицы, иначе на линии PA1 логический ноль. Для визуального отображения сигнала на линии воспользуемся цветным пробником логических уровней 0 и 1, который в программе Proteus представлен компонентами LOGICPROBE и LOGICPROBE (BIG) из раздела Logic Probes библиотеки Debugging Tools (рис. 17б). В нашем примере для контроля входного сигнала на линии PA1 микроконтроллера добавим в рабочее поле проекта компонент LOGICPROBE (BIG) и подключим его выход к выводу PA1 так, как показано на рис. 18. В результате при появлении на входе PA1 значения логической единицы пробник будет подсвечен красным цветом, при появлении же значения логического нуля пробник будет подсвечен синим цветом. Также на пробнике визуально отображаются значения 0 и 1. Напряжение, выдаваемое датчиком при нажатии (VOUT when touching), – в нашем примере 5 В – указывают в поле Advanced Properties окна его свойств (рис. 19), которое открывают щелчком правой кнопки мыши по пиктограмме датчика на схеме контекстного меню и выбором в нём команды Edit Properties.

Для работы с буквенно-цифровым дисплеем в CodeVisionAVR предусмотрена библиотека alcd.h, которая содержит функции вывода символов на экран дисплея, среди которых:

- void char lcd_init(unsigned char lcd_columns) – инициализация буквенно-цифрового дисплея, очистка экрана и установка курсора в позицию 0, 0. Параметр lcd_columns – количество столбцов дисплея;
- void lcd_clear(void) – очистка экрана и установка курсора в позицию 0, 0;
- void lcd_gotoxy(unsigned char x, unsigned char y) – установка курсора в позицию x, y экрана, где x – это номер столбца, y – номер строки;
- void lcd_putchar(char c) – вывод символа c на экран в текущую позицию курсора;
- void lcd_puts(char *str) – функция выводит строку str, расположенную в SRAM на экран, начиная с текущей позиции курсора;
- void lcd_putsf(char flash *str) – функция выводит строку str, расположенную во FLASH на экран, начиная с текущей позиции курсора;
- void lcd_putse(char eeprom *str) – функция выводит строку str, расположенную в EEPROM на экран, начиная с текущей позиции курсора.

Применение функций рассмотрим на примере вывода на экран буквенно-цифрового дисплея графической и текстовой информации, формирующих предупреждающие сообщения, оповещающие о нажатии датчика касания.

Текст программы:

```
#include <mega16.h> // подключение заголовочных файлов
#include <alcd.h> // в которых содержатся
#include <stdio.h> // прототипы функций
#include <delay.h>
```

```
  #include <math.h>
  #include <stdlib.h>
  // функция отрисовки рамки на экране буквенно-цифрового дисплея
  void ramka()
  { int i;
  lcd_clear( ); // очистка экрана дисплея
  i=0;
  // отрисуем верхнюю сторону рамки
  while( 1 ) // выполняем цикл, пока не будет достигнуто
  // условие i==20 (ширина рамки 20 знакомест)
  { i++; // устанавливаем курсор на следующую позицию нулевой строки экрана
  lcd_gotoxy( i-1,0 );
  lcd_putchar( 0xff ); // выводим на экран символ заполненного знакоместа,
  // 0xff - код символа из таблицы кодов символов
  if( i==20 ){ break; } // выход из цикла while, если курсор достиг 19 столбца нулевой строки
  }
  delay_ms(40); // задержка 40 мс
  // отрисуем левую боковую сторону рамки
  lcd_gotoxy( 0,1 );
  lcd_putchar( 0xff );
  lcd_gotoxy( 0,2 );
  lcd_putchar( 0xff );
  delay_ms(40);
  i=0;
  // отрисуем нижнюю сторону рамки
  while( 1 ) // выполняем цикл, пока не будет достигнуто условие i==20
  {i++; // устанавливаем курсор на следующую позицию третьей строки экрана
  lcd_gotoxy( i-1,3 );
  lcd_putchar( 0xff ); // выводим на экран символ заполненного знакоместа 0xff
  if( i==20 ){ break; } // выход из цикла while, если курсор достиг 19 столбца третьей строки
  }
  delay_ms(40); // задержка 40 мс
  // отрисуем правую боковую сторону рамки
  lcd_gotoxy( 19,1 );
  lcd_putchar( 0xff );
  lcd_gotoxy( 19,2 );
  lcd_putchar( 0xff );
  delay_ms(40);
  }

  void main(void) // основная функция программы
```

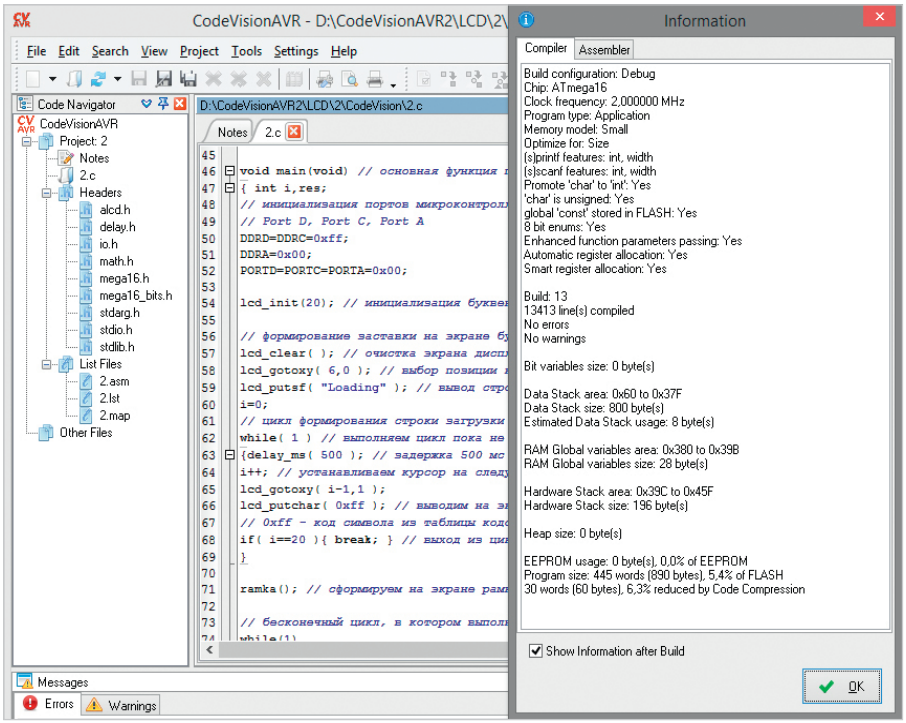


Рис. 20. Результат компиляции программного кода индикации состояния датчика касания в CodeVisionAVR

```
  { int i, res;
  // инициализация портов микроконтроллера
  // Port D, Port C, Port A
  DDRD=DDRC=0xff;
  DDRA=0x00;
  PORTD=PORTC=PORTA=0x00;

  lcd_init(20); // инициализация буквенно-цифрового дисплея

  // формирование заставки на экране буквенно-цифрового дисплея
  lcd_clear( ); // очистка экрана дисплея
  lcd_gotoxy( 6,0 ); // выбор позиции курсора (6 столбец, 0 строка)
  lcd_putsf(«Loading»); // вывод строки, начиная с текущей позиции курсора
  i=0;
  // цикл формирования индикатора загрузки
  while( 1 ) // выполняем цикл, пока не будет достигнуто условие i==20
  {delay_ms( 500 ); // задержка 500 мс
  i++; // устанавливаем курсор на следующую позицию первой строки экрана
  lcd_gotoxy( i-1,1 );
  lcd_putchar( 0xff ); // выводим на экран символ заполненного знакоместа,
  // 0xff - код символа из таблицы кодов символов
  if( i==20 ){ break; } // выход из цикла while, если курсор достиг 19 столбца первой строки
  }
  ramka(); // сформируем на экране рамку

  // бесконечный цикл, в котором выполняется опрос датчика касания
  while(1)
  {
  // запись сигнала, полученного с линии PA1 микроконтроллера, в переменную
  res=PINA.1;
  if (res==1) { // если на линии PA1 логическая единица
  lcd_gotoxy( 6,1 ); // выбор позиции курсора (6 столбец, 1 строка) на экране дисплея
  lcd_puts(«WARNING!»); // вывод предупреждающего сообщения на экран
  delay_ms(40); // задержка 40 мс
  }
  else // иначе, если на линии PA1 логический ноль
  {
  lcd_gotoxy( 5,1 ); // выбор позиции курсора (5 столбец, 1 строка)
  lcd_puts(«___OKey___»); // вывод сообщения на экран
  delay_ms(40); // задержка 40 мс
  }}
  }
```

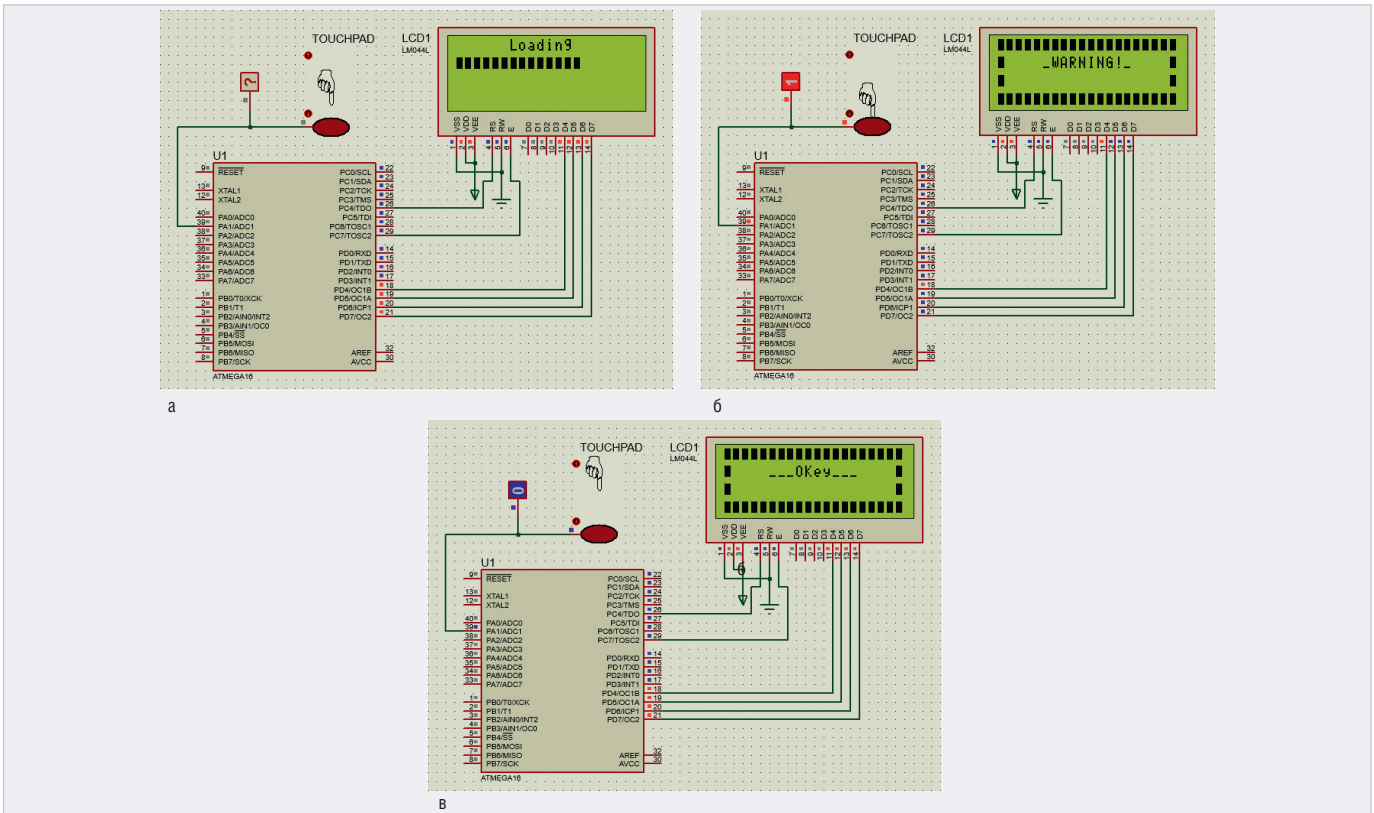


Рис. 21. Результат работы программы индикации состояния датчика TOUCHPAD в Proteus: (а) отображение анимированной заставки в виде строки загрузки на экране дисплея, (б) предупреждающие сообщения на экране буквенно-цифрового дисплея, когда сенсорная кнопка датчика касания нажата или (в) отжата

Введём текст программы в окне кода CodeVisionAVR и запустим командой основного меню Project/Build All компиляцию, по окончании которой выдаётся отчёт о наличии ошибок в коде программы (рис. 20). Если ошибки не обнаружены, на диске компьютера будет создан исполняемый файл для записи в микроконтроллер. Теперь перейдём в Proteus и в окне свойств микросхемы ATmega16 укажем путь к файлу прошивки на диске компьютера (в нашем примере к файлу с расширением .hex). Командой основного меню Debug/Run Simulation запустим симуляцию собранной схемы, результат которой представлен на рис. 21 (а, б, в), и проанализируем её работу.

После запуска моделирования программа микроконтроллера выполняет инициализацию его портов и буквенно-цифрового дисплея. Затем на экране дисплея отображается заставка в виде индикатора загрузки (рис. 21а). Формирование индикатора осуществляется последовательным заполнением первой строки экрана дисплея в позициях 0...19 символами ■ – заполненное знакоместо (шестнадцатиричный код такого символа – 0xff). В нулевой строке экрана отображается надпись Loading, вывод которой осуществляется с помощью функций lcd_gotoxy (выбор позиции курсора на экране) и lcd_putsf (вывод строки, начиная с текущей позиции курсора). Затем дисплей очищается, и на экране отображается рамка. Команды формирования рамки собраны в отдельной функции ramka( ), инициализация которой выполнена в начале программы. Далее запускается цикл while(1), где непрерывно ведётся опрос линии PA1 микроконтроллера, подключённой к выводу данных датчика касания, и обработка двух условий. Для каждого условия происходит формирование оповещающего сообщения (WARNING! или ___OKey___), которое отображается на экране дисплея. Появление на линии логической единицы (истинно первое условие) означает, что сенсорная кнопка датчика нажата (рис. 21б). Когда на линии логический ноль (истинно второе условие) – кнопка датчика отжата (рис. 21в).

Проанализировав способы формирования программного кода управления буквенно-цифровым дисплеем в представленных выше примерах, можно прийти к выводу, что применение специальных инструментов (таких как генератор кода CodeWizardAVR и функции стандартных библиотек) расширяет возможности программы CodeVisionAVR, облегчает написание и позволяет существенно сократить объём кода программы инициализации микроконтроллера и время её выполнения.

Литература

1. Proteus VSM Help // Labcenter Electronics, 2020.
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.
5. *Евстифеев А.В.* Микроконтроллеры AVR семейства Mega. Руководство пользователя. М.: Издательский дом «Додэка-XXI», 2007.
6. *Хартов В.Я.* Микроконтроллеры AVR. Практикум для начинающих. М.: Издательство МГТУ им. Н.Э. Баумана, 2007.