Работа с последовательным интерфейсом I²C в программной среде Proteus 8.11

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассматривается проектирование схем микроэлектронных устройств с использованием интерфейса I²C в Proteus на примере его реализации в устройстве измерения температуры, собранном на основе микроконтроллера AVR (семейства AT90) и датчиков LM75AD. Описаны особенности написания программного кода на языке С для инициализации интерфейса и работы с ним. Приведён пример моделирования схемы, в которой проводится передача данных через I²C между датчиками температуры, сконфигурированными как ведомые устройства, и ведущим микроконтроллером AT90S8515, компиляция программы инициализации которого выполнена в CodeVisionAVR. Выполнено отображение принятых ведущим устройством данных на экране буквенно-цифрового дисплея LM016L. С помощью осциллографа проведён контроль сигналов, присутствующих на линиях SDA и SCL интерфейса I²C.

Введение

Двухпроводный последовательный интерфейс I²C (Integrated Circuit) обеспечивает взаимодействие микроконтроллера с множеством микросхем (энергонезависимой памятью, контроллерами параллельных портов, LCD-дисплеями, микроконтроллерами и различными специализированными устройствами). Данный интерфейс позволяет объединить до 128 устройств по схеме, приведённой на рис. 1.

Интерфейс представляет собой две линии: одна (SDA) используется для передачи данных, другая (SCL) – для тактовых сигналов. Через резисторы R1, R2 обе линии подключены к источнику питания VCC. Выходы устройств выполнены по схеме с открытым коллектором (стоком), что позволяет реализовать функцию «монтажное И» для выходных сигналов. Низкий уровень сигнала (логический 0) на выходе любого из устройств устанавливает низкий уровень на всей линии. Высокий уровень на линии устанавливается, когда выводы всех устройств находятся в третьем (высокоимпедансном) состоянии.

Устройство, подключённое к шине, может иметь статус ведущего (master) или ведомого (slave). Статус микроконтроллера устанавливается программно.

Протоколом работы шины предусмотрены:

- посылка ведущим устройством стартового бита начала обмена;
- передача последовательности из семи разрядов адреса ведомого устройства;
- транзакция чтения или записи 8-битовых данных;
- получение ведущим устройством битов подтверждения передачи адреса и данных;
- формирование бита подтверждения после приёма данных;
- посылка ведущим устройством стопового бита.

Шина I²C является последовательной: все данные и адреса передаются по линии SDA поразрядно. Каждый передаваемый бит сопровождается тактовым сигналом на линии SCL. В течение всего времени действия сигнала SCL (SCL = 1) состояние линии SDA должно оставаться неизменным. Изменение данных на линии SDA происходит при отсутствии тактового сигнала на линии SCL (SCL = 0). Исключение составляют стартовый и стоповый биты, определяющие начало и конец обмена. Стартовый бит формируется путём изменения уровня сигнала на линии SDA с 1 на 0 при SCL = 1, стоповый бит – при изменении сигнала SDA с 0 на 1 также при SCL = 1.

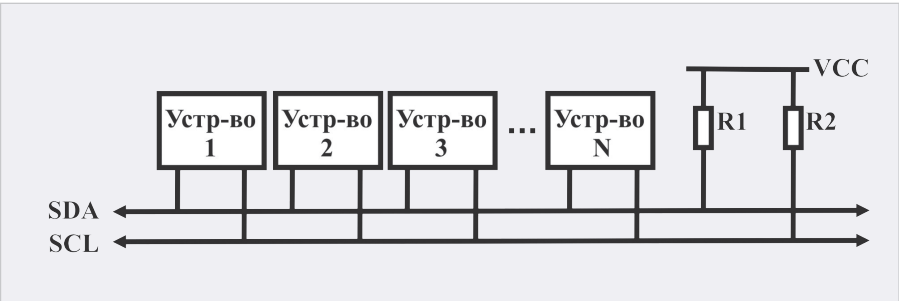

Рис. 1. Схема соединения устройств по интерфейсу I²C

Протокол обмена по шине предполагает передачу двух типов кадров: с адресом и данными. Кадр 7-разрядного адреса содержит:

- S – стартовый бит;
- A – 7-разрядный адрес ведомого устройства, передаваемый ведущим, начиная со старшего разряда;
- R/W – управляющий бит, определяющий тип транзакции на шине (R/W = 0 – запись, R/W = 1 – чтение);
- ACK – бит подтверждения.

Адрес, передаваемый ведущим устройством после захвата шины, поступает ко всем устройствам, подключённым к ней. Каждое из устройств сравнивает поступающий адрес с собственным. При распознавании ведомым устройством своего адреса оно возвращает на линию SDA сигнал подтверждения ACK низкого уровня во время 9-го тактового сигнала SCL. Если по каким-либо причинам ведомое устройство не способно обслужить запрос ведущего, оно удерживает на линии SDA сигнал высокого уровня. Нулевой адрес используется для общего вызова всех устройств. Управляющий бит в этом случае устанавливается в 0, чтобы обеспечить передачу одного и того же сообщения всем устройствам.

После передачи адреса начинается передача данных. Кадр байта данных содержит восемь битов данных и один бит подтверждения, формируемый приёмником. Данные, так же как и адрес, передаются последовательно бит за битом, начиная со старшего разряда. После приёма каждого байта данных приёмник вырабатывает сигнал подтверждения ACK путём выдачи на линию SDA сигнала низкого уровня. Высокий уровень сигнала свидетельству-

ет об ошибке или невозможности продолжить приём. Не получив подтверждения от приёмника, ведущий может прекратить передачу данных, сформировав сигналы состояния Stop.

После завершения передачи байта данных работа может быть продолжена либо для передачи следующего байта в том же направлении без изменения ведомого, либо выбором нового ведомого и/или сменой направления обмена. При завершении обмена шина освобождается ведущим устройством.

Создание в Proteus схемы передачи данных по I²C между датчиками LM75AD и микроконтроллером AVR

В микроконтроллерах AVR протокол обмена по шине I²C выполняется программно или программно-аппаратно. Программный способ осуществляется с использованием библиотеки функций и применяется в микроконтроллерах семейств ATtiny и AT90, в которых отсутствуют встроенные аппаратные средства, реализующие протокол обмена. Программная реализация протокола I²C для разных случаев взаимодействия устройств представляет собой набор программ, эмулирующих работу ведущего и ведомых устройств с учётом функциональных требований. Наиболее сложным является случай, когда в качестве ведущего и ведомых выступают микроконтроллеры, которые могут быть как передатчиками, так и приёмниками при обмене данными и не имеют встроенных средств обмена по I²C. Более простой случай, когда одно из устройств (микроконтроллер) является ведущим, а ведомые устройства (датчики, устройства памяти, часы реального времени и др.) содержат встроенный порт для обмена по I²C.

Рассмотрим работу с шиной I²C в Proteus на примере схемы термометра, реализованной на микросхемах LM75AD (датчики температуры) и микроконтроллере AT90S8515. Измеренная температура отображается на экране буквенно-цифрового дисплея LM016L.

Библиотека программной среды Proteus содержит как аналоговые, так и цифровые компоненты, а также устройства вывода информации и микроконтроллеры с возможностью их программирования. Большой набор инструментов и функций, среди которых вольтметр, амперметр, осциллограф, генераторы сигналов, а также возможность отлаживать программное обеспечение микроконтроллеров, делают

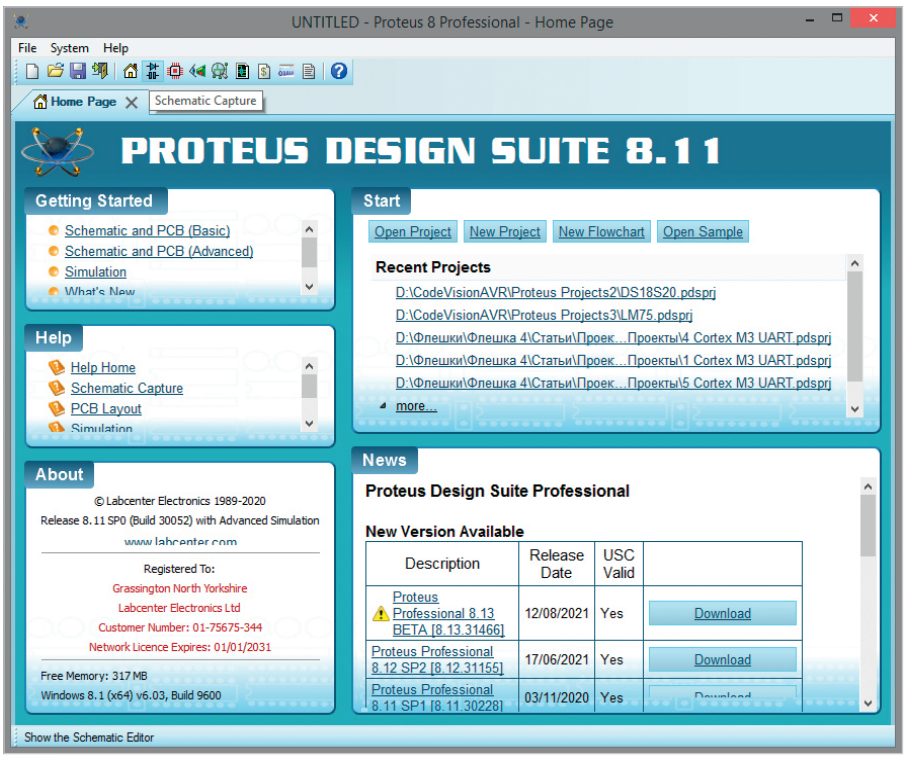

Рис. 2. Стартовое окно программной среды Proteus

Proteus мощным средством разработки электронных устройств, позволяя реализовать на персональном компьютере виртуальную лабораторию, в которой можно максимально приближённо имитировать реальные лабораторные условия, как с точки зрения элементной базы, так и современных приборов. Proteus 8.11 объединяет две основных программы: Schematic Capture – средство разработки и отладки в режиме реального времени электронных схем и PCB Layout – средство разработки печатных плат.

При проектировании устройства измерения температуры, работающего под управлением микроконтроллера AVR, написание программы инициализации и её компиляцию удобно выполнить с помощью CodeVisionAVR 3.12 (интегрированной среды разработки программного обеспечения для микроконтроллеров семейства AVR фирмы Atmel, которая имеет в своём составе компилятор языка C для AVR). В таком случае проект схемы электрической принципиальной создают без использования мастера – при помощи кнопки Schematic Capture верхней панели инструментов стартового окна Proteus (рис. 2). Нажатие кнопки открывает новую одноимённую вкладку, в рабочем поле которой и будет выполняться разработка схемы.

Соберём схему на основе микроконтроллера AT90S8515, буквенно-цифрового дисплея LM016L и трёх датчиков температуры LM75AD, для чего добавим её компоненты в рабочую область редактора Schematic Capture и соединим их так, как показано на рис. 3. Выбор компонентов из базы данных для последующего их размещения в рабочей области программы 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 схемного редактора.

Для добавления микросхемы микроконтроллера (рис. 4а) в рабочее поле проекта в левой верхней части окна Pick Devices в поле Category щелчком левой кнопки мыши выбирают из списка библиотеку Microprocessor ICs. Пакет Microprocessor ICs позволяет включать в эмуляцию смешанной схемы определённые микроконтроллеры с возможностью написания и отладки программного кода. В поле Sub-category таким же способом задают семейство микроконтроллеров выбранной библиотеки (в нашем примере AVR Family). Все компоненты семейства отображаются в поле Showing local results. В поле Manufacturer выбирают производителя микроконтроллера. Если производитель не имеет значения – указывают значение All Manufacturers. Для ускорения поиска компонентов можно воспользоваться строкой фильтра

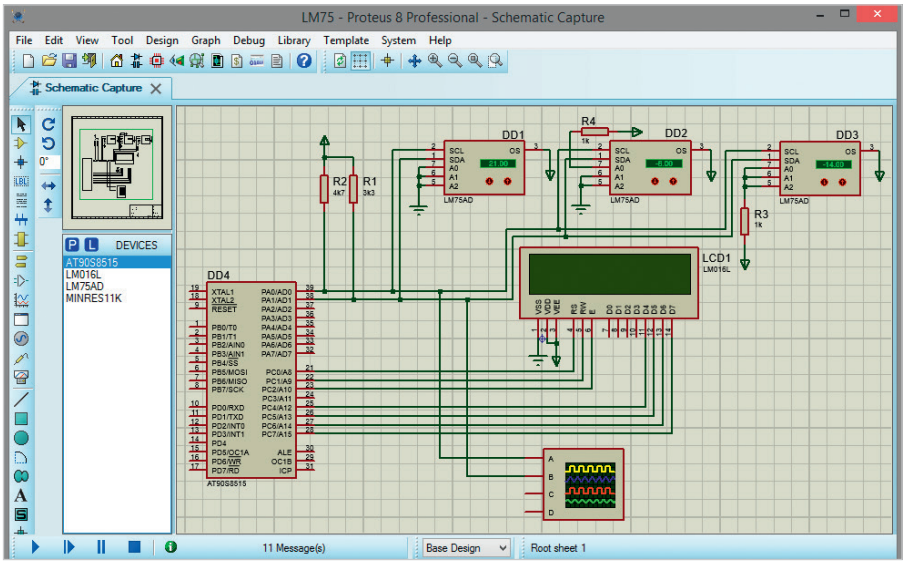

Рис. 3. Подключение к микроконтроллеру AT90S8515 устройства вывода информации и группы датчиков LM75AD по интерфейсу I²C в рабочей области редактора Schematic Capture программы Proteus

Keywords, которая расположена в верхнем левом углу окна Pick Devices. После выбора микроконтроллера (в нашем примере это микросхема AT90S8515) его условное графическое обозначение отобразится в поле предварительного просмотра Preview. Посадочное место компонента будет показано в поле PCB Preview. Если для микроконтроллера назначено несколько посадочных мест, то все возможные варианты будут доступны для выбора из выпадающего меню, которое расположено под полем PCB Preview. Для размещения микроконтроллера на схеме нажимают на кнопку OK, после чего окно Pick Devices будет закрыто, а символ компонента прикреплён к курсору мыши, при помощи которого его помещают в нужное место на схеме щелчком левой кнопки мыши.

Аналогичным образом добавим в рабочее поле проекта из раздела 0.6W Metal Film библиотеки Resistors четыре резистора MINRES11K (рис. 4б), а затем микросхему буквенно-цифрового дисплея LM016L [3], которая находится в разделе Alphanumeric LCDs библиотеки Optoelectronics (рис. 4в).

Микросхема LM016L 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;

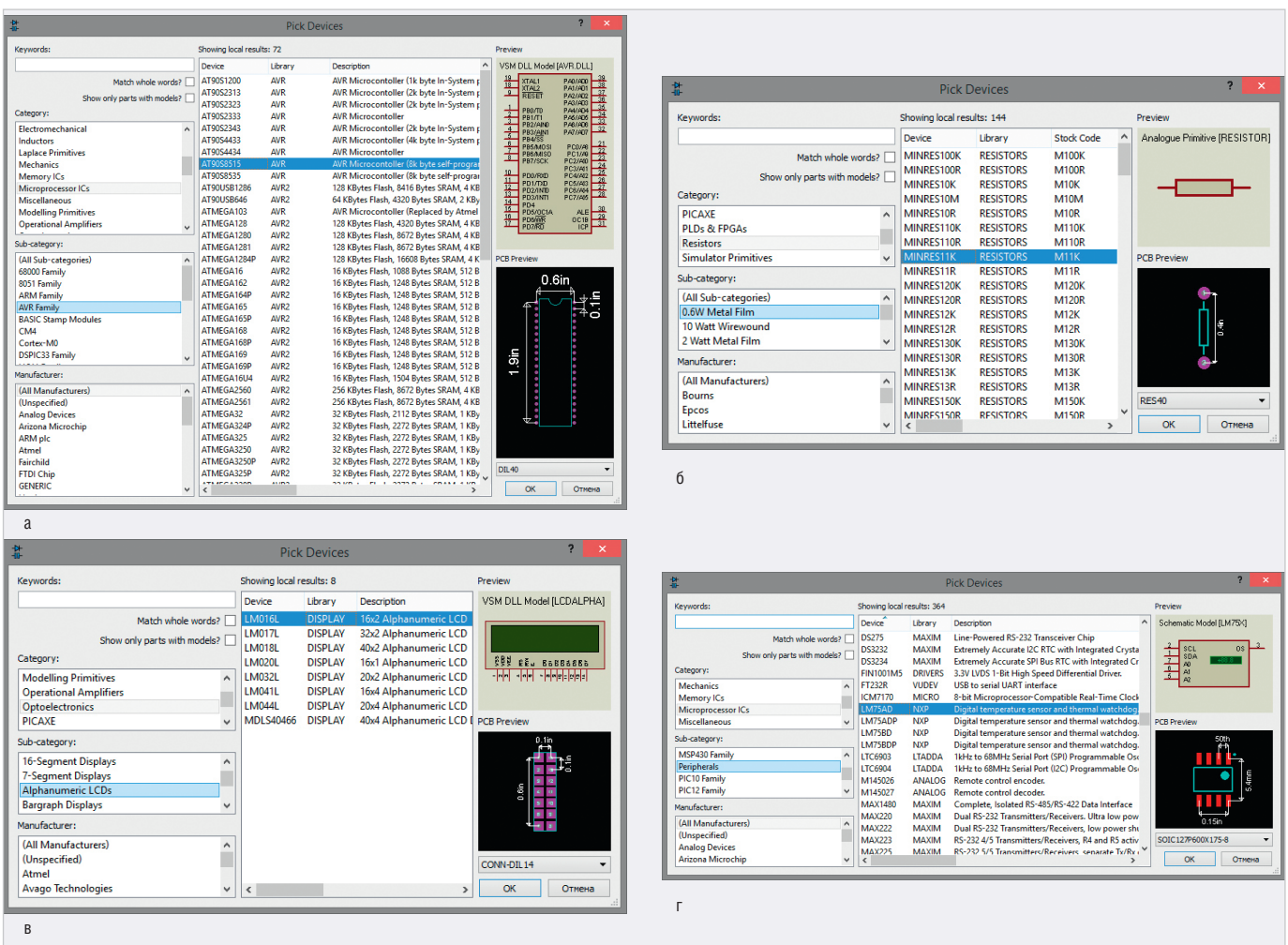

Рис. 4. Раздел: (а) AVR Family библиотеки Microprocessor ICs, (б) 0.6W Metal Film библиотеки Resistors, (в) Alphanumeric LCDs библиотеки Optoelectronics, (г) Peripherals библиотеки Microprocessor ICs

- D0...D7 – шина данных/команд.

Микросхема LM016L может работать в двух режимах:

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

В представленном примере вывод данных на экран дисплея разрешением 16 символов на 2 строки выполнен в 4-разрядном режиме.

Для подключения микросхемы LM016L 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 D4...D7 к выводам PC4...PC7, а выводы RS, RW и E к выводам PC0...PC2 микроконтроллера AT90S8515 так, как показано на рис. 3.

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 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 (рис. 5) строки GROUND и POWER. Панель открывают нажатием кнопки Terminals Mode на левой панели схемного редактора. Выбор линий портов микроконтроллера для подключения к указанным выводам дисплея выполняется разработчиком 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 Clock Frequency (тактовая частота) – рис. 6а, значение которой должно совпадать с частотой работы микроконтроллера (в нашем примере 3,6864 МГц).

Из раздела Peripherals библиотеки Microprocessor ICs добавим в рабочую область проекта три датчика температуры (рис. 4г), разместим их так, как показано на рис. 3, и подсоединим их выводы SCL и SDA к выводам PA0 и PA1 микроконтроллера, а выводы OS к символу питания.

Для каждого датчика в окне его свойств (окно открывают командой Edit Properties

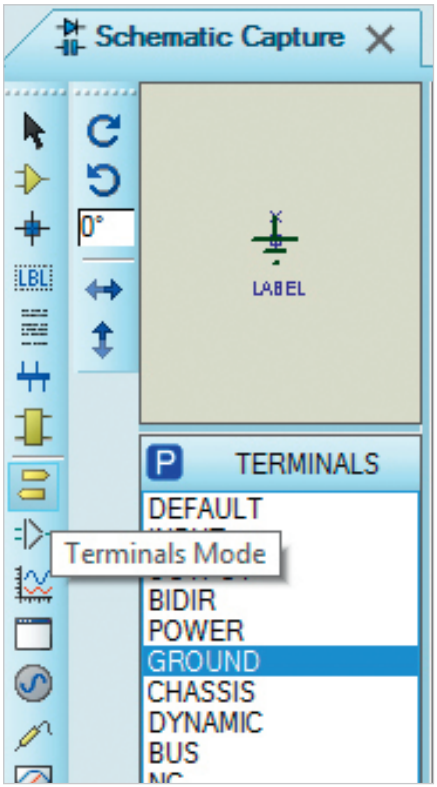

Рис. 5. Открытие с помощью кнопки Terminals Mode панели TERMINALS и выбор символа «земли»

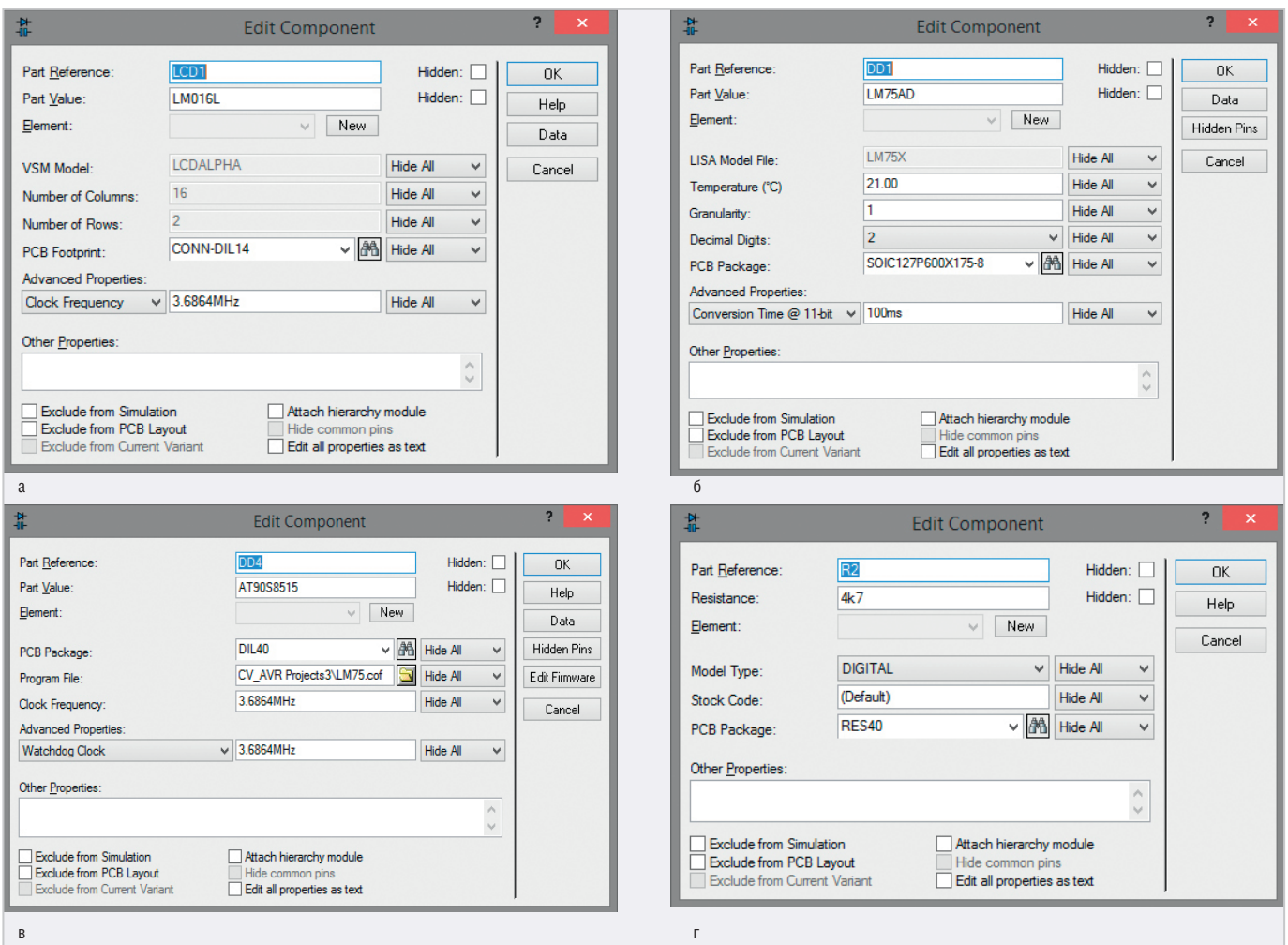

Рис. 6. Окно свойств: (а) микросхемы LM016L, (б) датчика температуры LM75AD, (в) микроконтроллера AT90S8515, (г) резистора

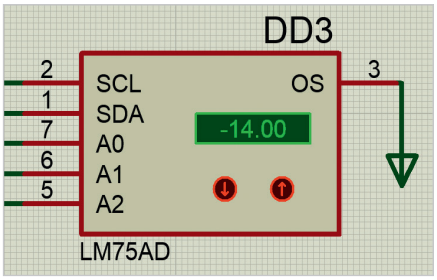

Рис. 7. Приближённый вид датчика температуры LM75AD на схеме

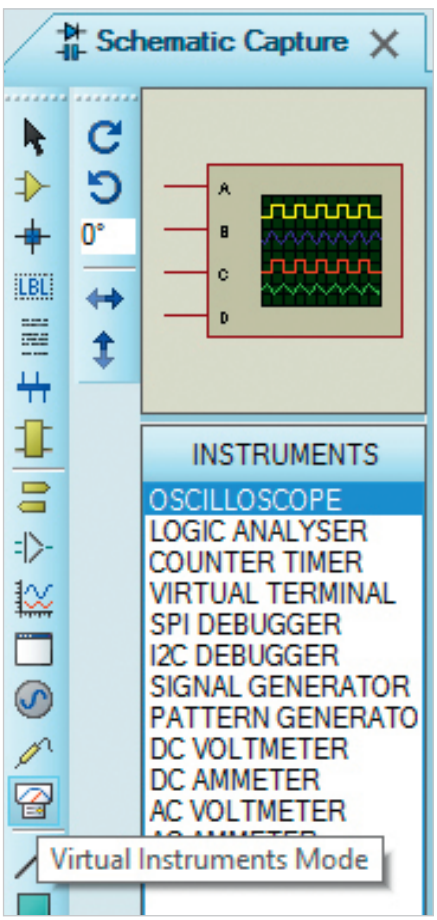

Рис. 8. Открытие с помощью кнопки Virtual Instruments Mode панели INSTRUMENTS и выбор четырёхканального осциллографа

контекстного меню, которое вызывают щелчком правой кнопки мыши по пиктограмме датчика на схеме) определим следующие параметры (рис. 6б):

- текущее значение температуры (поле Temperature (°C)) – в нашем примере 21,00; –6,00; –14,00 °C;
- точность измерения (поле Granularity) – 1 °C;
- количество знаков после запятой (поле Decimal Digits) – 2.

С помощью переключателей «↑» и «↓», размещённых на пиктограмме (рис. 7), температуру датчика можно подстраивать на схеме.

Датчик LM75AD (цифровой температурный датчик с интерфейсом I²C) имеет восемь контактов, назначение которых следующее:

- GND – «земля»;
- A0, A1, A2 – линии, задающие адрес устройства;
- OS – сигнал превышения заданного порога температуры;
- SDA, SCL – линии шины I²C;
- VCC – напряжение питания +5 В.

Линии питания и «земли» на пиктограмме датчика в Proteus отсутствуют.

Микросхема LM75AD содержит встроенную оперативную память и схему слежения для контроля выхода температуры за установленное пользователем пороговое значение. Взаимодействие микросхемы с микроконтроллером обеспечивает двухпроводный последовательный интерфейс I²C, который представляет собой две линии: одна (SDA) используется для передачи данных, другая (SCL) – для тактовых сигналов. В нашем примере для шины I²C используются линии PA0 и PA1 микроконтроллера. Через резисторы R1 (3,3 кОм), R2 (4,7 кОм) обе линии подключены к источнику питания +5 В.

Slave-адрес каждого устройства шины I²C представляет собой байт, где четыре старших бита являются идентификатором типа устройства. Значение следующих трёх битов является адресом устройства. Самый младший бит slave-адреса определяется тем, что хочет делать основное устройство master – читать или записывать. При чтении этот бит – 1, при записи – 0. Часть slave-адреса A0...A2 является адресом устройства, для определения которого его соответствующие выводы A0, A1 и A2 подключают к «земле» или через подтягивающие резисторы ёмкостью 1 кОм к напряжению +5 В.

В нашем примере к шине I²C подключено три ведомых устройства (датчики температуры). Комбинация сигналов на выводах A0...A2 определяет номер датчика на шине. Датчик DD1 определён как ведомое устройство под номером 0 (все его адресные выводы подключены к «земле»). Датчикам DD2 и DD3 присвоены номера 1 и 7, что определено комбинацией сигналов 100 (вывод A0 подключён к напряжению +5 В, а выводы A1 и A2 – к «земле») и 111 (все адресные выводы подключены к напряжению +5 В) на выводах A0...A2. После передачи адреса начинается передача данных.

Для настройки параметров микроконтроллера откроем окно его свойств (рис. 6в) и укажем путь к файлу hex (или cof) на диске компьютера (поле Program File), значение тактовой частоты микроконтроллера (поле Clock Frequency) – 3,6864 МГц и частоты сторожевого таймера Watchdog Clock (поле Advanced Properties) – 3,6864 МГц. В окне настроек резисторов (рис. 6г) в поле Resistance укажем их сопротивление (для R1 – 3,3 кОм, для R2 – 4,7 кОм, для R3, R4 – 1 кОм).

Для исследования работы интерфейса I²C применим такой виртуальный инструмент программы Proteus, как четырёхканальный осциллограф, каналы A и B которого подключим к выводам PA0, PA1 микроконтроллера так, как показано на рис. 3, для чего выберем строку OSCILLOSCOPE на панели INSTRUMENTS (рис. 8) и разместим прибор при помощи мыши в рабочем поле программы. Чтобы открыть панель INSTRUMENTS, на левой панели инструментов схемного редактора нажмём на пиктограмму Virtual Instruments Mode.

В Proteus осциллограф имеет четыре сигнальных входа (каналы A, B, C и D) и может отображать осциллограммы четырёх сигналов одновременно. Осциллограф заземлён по умолчанию, поэтому вывод заземления отсутствует. Также отсутствует и вывод внешней синхронизации. Лицевая панель прибора открывается вследствие запуска симуляции схемы. Пиктограмма используется для подключения прибора к схеме, в свою очередь, лицевая панель предназначена для настройки прибора и наблюдения формы исследуемых сигналов. В левой части лицевой панели четырёхканального осциллографа расположен графический дисплей, который предназначен для графического отображения формы сигнала, а именно: для отображения напряжения по вертикальной оси и, соответственно, времени по горизонтальной оси. Также прибор оснащён курсорами для проведения измерений во временной области, которые при необходимости можно перемещать при помощи левой кнопки мыши. Добавление курсоров становится возможным после нажатия на кнопку Cursors в окне Trigger панели управления осциллографа, которая находится в правой части его лицевой панели и предназначена для настройки отображения измеряемого сигнала. На панели управления размещено шесть окон настроек:

- Trigger (Синхронизация);
- Channel A (Канал A);
- Channel C (Канал C);
- Channel B (Канал B);
- Channel D (Канал D);
- Horizontal (Развёртка).

В нижней части окна Channel A расположена ручка, при помощи которой зада-

ётся величина деления по оси Y (количество вольт на деление). Начальная точка вывода сигнала на оси Y указывается в поле Position. Поле может принимать как положительное, так и отрицательное значение. Выбор положительного значения в данном поле сдвигает начальную точку вверх по оси Y, соответственно выбор отрицательного значения сдвигает начальную точку вниз. Выбор режима работы осуществляется посредством установки ползунка в одну из четырёх позиций: AC, DC, GND, OFF. В режиме AC отображается только переменная составляющая сигнала. В режиме DC отображается сумма переменной и постоянной составляющих сигнала. В случае выбора позиции GND входной канал замыкается на землю, а на экране графического дисплея отображается прямая линия в точке исходной установки оси Y. Установка ползунка в позицию OFF выключает отображение сигнала на дисплее. Также в окне Channel A расположено две кнопки:

- Invert – задаёт инверсный режим работы осциллографа, в котором сигнал инвертируется относительно положения нуля;
- A+B – задаёт режим, в котором на экране графического дисплея отображается суммарный сигнал каналов A и B.

Интерфейс окон Channel C, Channel B, Channel D аналогичен уже рассмотренному окну Channel A за исключением того, что в окне Channel C вместо кнопки A+B присутствует кнопка C+D, задающая режим, в котором на экране графического дисплея отображается суммарный сигнал каналов C и D. В окнах Channel B и Channel D такая кнопка вообще отсутствует.

В нижней части окна Horizontal расположена ручка, при помощи которой задаётся величина деления по оси X. Начальная точка вывода сигнала на оси X указывается в поле Position. Поле может принимать как положительное, так и отрицательное значение. Отображение сигнала на экране графического дисплея производится слева направо. Выбор положительного значения в данном поле сдвигает начальную точку вывода сигнала вправо, соответственно выбор отрицательного значения сдвигает начальную точку влево. Выбор режима развёртки осуществляется в поле Source посредством установки ползунка в одну из следующих позиций: ∧, A, B, C, D. В случае выбора режима ∧ (сигнал по оси Y/время) на экране графического дисплея по оси Y будут отображаться сигналы каналов A, B, C, D, а ось X будет осью времени. Режимы A, B, C, D – это режимы наблюдения фигур Лиссажу. Выбор такого режима может быть полезен для изучения фаз сигналов.

В верхней левой части панели управления осциллографа размещено окно Trigger (Синхронизация). Выбор канала для запуска синхронизации производится в поле Source посредством установки ползунка в одну из следующих позиций: A, B, C, D. Осуществить выбор запуска сигнала синхронизации – по фронту или по срезу – можно в соответствующем поле посредством установки ползунка в одну из позиций.

В правой части окна Trigger находятся кнопки выбора режима синхронизации:

- One-Shot (Однократный) – режим ожидания сигнала синхронизации (используется для регистрации однократного сигнала);
- Auto (Автоматический) – запуск осциллограммы производится автоматически при подключении осциллографа к схеме и включении эмуляции схемы.

Результаты работы четырёхканального осциллографа отображаются на экране графического дисплея, расположенном в левой части лицевой панели данного прибора, в виде кривых, которые представляют входные сигналы, полученные с входов A, B, C, D. В нашем примере осциллограф применён для отображения сигналов SDA и SCL интерфейса I^2C.

После создания схемы, подключения всех приборов и настройки их параметров переходят к следующему этапу разработки: написанию программного кода управления устройством с помощью CodeVisionAVR. Среда разработки поддерживает все базовые конструкции языка C, 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. В CodeVisionAVR имеется набор команд управления буквенно-цифровыми дисплеями и датчиками температуры, функции шины I^2C и доступа к памяти. Программные средства позволяют напрямую обращаться к регистрам микроконтроллера и управлять состоянием линий портов.

В результате компиляции программного кода управления устройством (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, который выполняют кнопкой Run the simulation, расположенной в левом нижнем углу окна редактора или командой основного меню Debug/Run Simulation. Временную приостановку процесса симуляции выполняют кнопкой Pause the simulation, or start up at time 0 if stopped (кнопка находится в левом нижнем углу окна редактора). Останавливают моделирование кнопкой Stop the simulation.

Создание программного кода в CodeVisionAVR

Формирование программного кода в CodeVisionAVR выполняют при помощи автоматического генератора CodeWizardAVR (для быстрого получения кода, который требует редактирования) или вручную с нуля, используя синтаксис языка программирования C и функции стандартных библиотек программы. Одна часть библиотек встроена в компилятор, другая (файлы с расширением .lib) располагается в поддиректории ..\LIB. Каждая библиотека представляет собой набор определённых функций, среди которых функции шины I^2C для работы с температурными датчиками LM75AD (библиотека lm75.lib) и LCD-функции для работы с дисплеем (библиотека lcd.lib). Для использования в программе библиотечных функций необходимо подключить с помощью директивы #include соответствующие заголовочные файлы (lm75.h – для работы с датчиком LM74AD, lcd.h – для работы с буквенно-цифровым дисплеем).

Функции протокола I^2C рассматривают микроконтроллер как master (ведущий) шины, а периферийные устройства – как slaves (ведомые). Их применяют для:

- инициализации шины I^2C (функция void i2c_init(void));
- чтения байта из шины (функция unsigned char i2c_read (unsigned char ack)). Параметр ack (подтверждение прочтения байта) устанавливают в 0, когда подтверждение не требуется, или в 1, когда должно быть выдано подтверждение после прочтения байта данных;
- записи байта data в шину (функция unsigned char i2c_write (unsigned char data)).

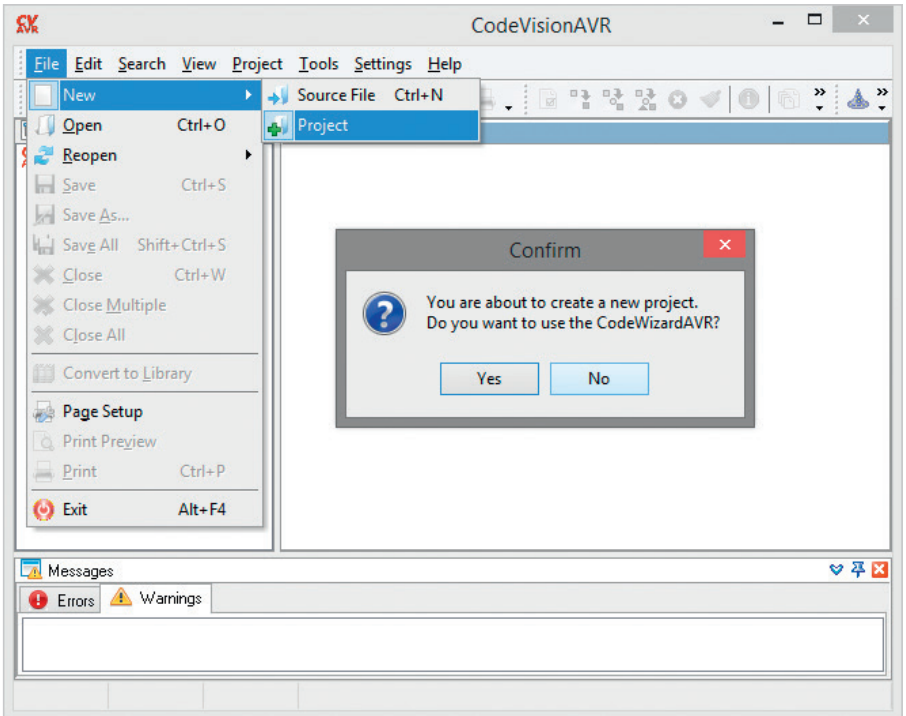

Рис. 9. Создание нового проекта в CodeVisionAVR

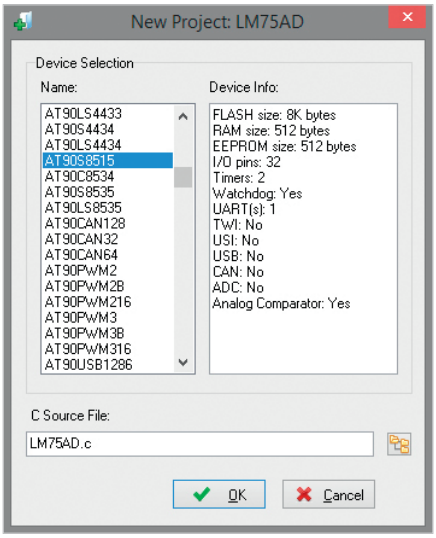

Рис. 10. Выбор микроконтроллера AT90S8515 в окне New Project

Прототипы этих функций размещаются в файле i2c.h, расположенном в поддиректории ..\INC. Перед их использованием в коде программы инициализации микроконтроллера необходимо объявить порт и биты микроконтроллера для связи через шину I^2C, а директивой #include подключить файл i2c.h.

Для работы с датчиком LM75AD применяют следующие функции:

- **void lm75_init** (unsigned char chip, signed char thyst, signed char tos, unsigned char pol) – функция инициализации датчика температуры, которую вызывают для всех имеющихся в проекте датчиков, подключённых к шине I^2C (их количество не должно превышать 8). В программе адрес датчиков (от 0 до 7) задают с помощью параметра chip, на схеме – с помощью подачи комбинации нулей и единиц на выводы A0...A2. Например, комбинацией сигналов 111 определяют седьмой подключённый к шине датчик. Вывод OS становится активным, когда температура, выраженная в °C, превышает предел tos и выходит из активного состояния, когда температура падает ниже предела thyst. Параметр pol представляет полярность выхода OS в активном состоянии. Если pol – 0, активным является низкий уровень, а если pol – 1, то высокий уровень. Прежде чем вызвать функцию инициализации датчика температуры, должна быть инициализирована шина I^2C;
- **int lm75_temperature_10** (unsigned char chip) – функция определения температуры датчика с адресом chip. Температура выражена в °C и умножена на 10.

Прототипы этих функций размещаются в файле lm75.h, который в программе подключают директивой #include. До подключения файла lm75.h необходимо объявить порт микроконтроллера и его биты для связи с датчиком LM75AD через шину I^2C.

Создание нового проекта в CodeVisionAVR

Новый проект в CodeVisionAVR создают командой основного меню File/New/Project. В процессе создания открывается диалоговое окно, где система предлагает воспользоваться генератором кода CodeWizardAVR, с помощью которого задают параметры микроконтроллера, его внутренних ресурсов и используемых в схеме периферийных устройств. 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, буквенно-цифровых и графических дисплеев и др. Однако в процессе работы мастера формируется достаточно объёмный код, который впоследствии приходится редактировать.

В нашем примере все настройки будут выполнены вручную программным способом, поэтому от запуска CodeWizardAVR можно отказаться, нажав на кнопку No (рис. 9). В результате будет открыто окно выбора директории размещения нового проекта Create New Project, где указывают имя проекта (поле «Имя файла») и его тип (поле «Тип файла»). Кнопкой «Сохранить» открывают окно New Project (рис. 10), где в поле Name выбирают микроконтроллер, под управлением которого работает собранная схема (его описание отобразится в поле Device Info). Далее нажатием кнопки OK открывают окно настройки параметров проекта CodeVisionAVR (Configure Project), переходят на вкладку C Compiler, на которой выбирают закладку Code Genereration (рис. 11), где указывают:

- размер стека данных в байтах (поле Data Stack Size) – для компиляции кода в нашем примере значения 128 будет достаточно;
- размер внутренней (поле Internal RAM Size) и внешней (поле External RAM Size) оперативной памяти – 512 и 0 байт соответственно;
- тактовую частоту микроконтроллера (поле Clock) – 3,6864 МГц;
- модель памяти (поле Memory Model) – Small.

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

Формирование и компиляция программного кода передачи по I^2C измеренной датчиками температуры

Для измерения с помощью датчиков LM75AD температуры и её отображения на экране буквенно-цифрового дисплея напишем программу инициа-

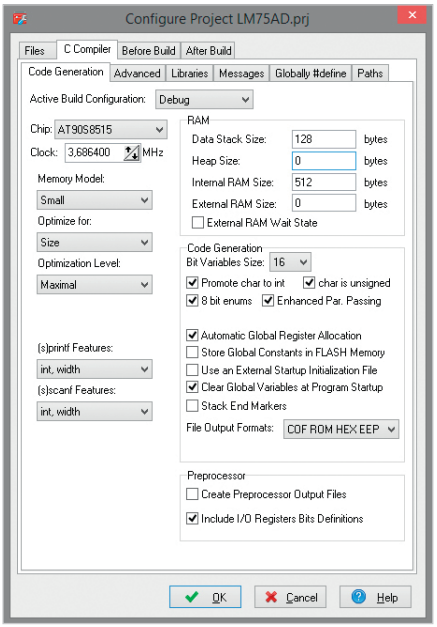

Рис. 11. Закладка Code Generation окна настройки параметров проекта CodeVisionAVR

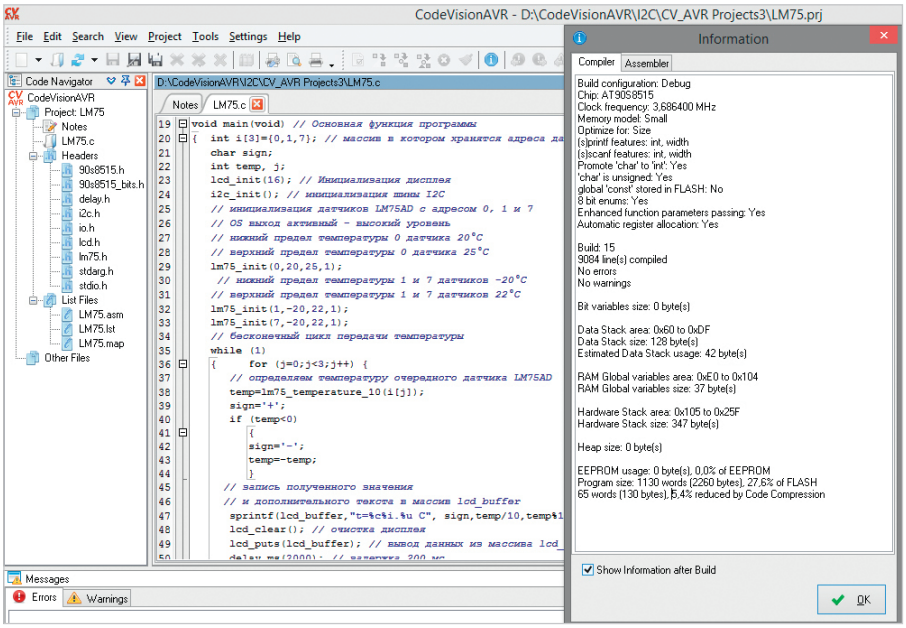

Рис. 12. Программа передачи по I²C измеренной датчиками LM75AD температуры в окне кода CodeVisionAVR и результат её компиляции

лизации микроконтроллера на языке С с применением стандартных функций CodeVisionAVR.

Текст программы:

```
#asm
.equ __i2c_port=0x1b // для связи через шину I2C используем
.equ __scl_bit=0 // 0 бит и
.equ __sda_bit=1 // 1 бит порта PA микроконтроллера AT90S8515
#endasm

#asm
.equ __lcd_port=0x15 // для подключения LCD-дисплея используем порт PC
#endasm

#include <lm75.h> // подключение заголовочных файлов
#include <stdio.h> // в которых содержатся
#include <90s8515.h> // прототипы функций
#include <delay.h>
#include <lcd.h>

char lcd_buffer[33]; // объявление массива данных для вывода на экран дисплея

void main(void) // основная функция программы
{ int i[3]={0,1,7}; // массив, в котором хранятся адреса датчиков
char sign;
int temp, j;
lcd_init(16); // инициализация дисплея
i2c_init(); // инициализация шины I2C
// инициализация датчиков LM75AD с адресом 0, 1 и 7
// OS выход активный - высокий уровень
// нижний предел температуры 0 датчика 20°C
// верхний предел температуры 0 датчика 25°C
lm75_init(0,20,25,1);
// нижний предел температуры 1 и 7 датчиков -20°C
// верхний предел температуры 1 и 7 датчиков 22°C
lm75_init(1,-20,22,1);
lm75_init(7,-20,22,1);
// бесконечный цикл передачи температуры
while (1)
{ for (j=0;j<3;j++) {
// определяем температуру очередного датчика LM75AD
temp=lm75_temperature_10(i[j]);
sign='+';
if (temp<0)
{
sign='-';
temp=-temp;
}
// запись полученного значения
// и дополнительного текста в массив lcd_buffer
sprintf(lcd_buffer,"t=%c%i.%u C", sign,temp/10,temp%10);
lcd_clear(); // очистка дисплея
lcd_puts(lcd_buffer); // вывод данных из массива lcd_buffer на экран дисплея
delay_ms(2000); // задержка 2000 мс
} } }.
```

Введём текст программы в окне кода CodeVisionAVR и запустим компиляцию (рис. 12). После чего перейдём в Proteus и в окне свойств микросхемы AT90S8515 укажем путь к файлу прошивки на диске компьютера. Запустим моделирование собранной схемы, результат которого представлен на рис. 13, и проанализируем её работу.

После запуска программа инициализации определяет биты порта микроконтроллера для связи через шину I²C. Для этого в программу включен ассемблерный код:

```
#asm
.equ __i2c_port=0x1b
.equ __scl_bit=0
.equ __sda_bit=1
#endasm,
```

в котором директива ассемблера .equ присваивает идентификатору __i2c_port значение, соответствующее адресу регистра PORTA порта PA микроконтроллера. Этот адрес прописан в строке **sfrb PORTA=0x1b**; находящегося в поддиректории \INC файла 90s8515.h. Идентификаторам __scl_bit и __sda_bit директивы .equ присваивают значения номеров соответствующих битов, которые будут использованы для передачи данных через линии SCL, SDA шины I²C. Директива #asm оповещает компилятор о начале ассемблерного кода, а директива #endasm – о его завершении.

В следующем блоке кода объявлен порт микроконтроллера для подключения LCD-дисплея:

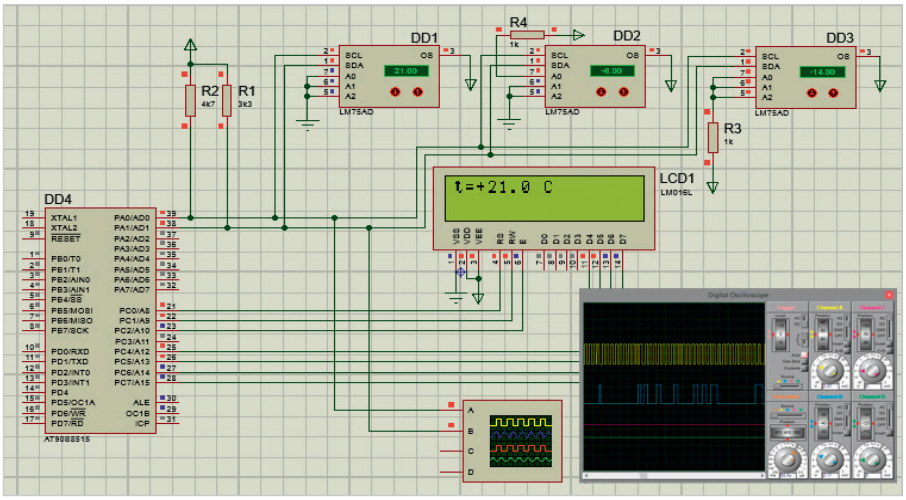

Рис. 13. Результат работы схемы измерения температуры с помощью датчиков LM75AD

```
#asm
.equ __lcd_port=0x15
#endasm.
```

Во второй строке кода директива ассемблера .equ присваивает идентификатору __lcd_port значение, соответствующее адресу регистра PORTC порта PC, которое при необходимости можно найти в строке **sfrb PORTC=0x15**; файла 90s8515.h.

При трансляции ассемблерного кода, полученного при компиляции данного проекта компилятором C CodeVisionAVR, ассемблер вместо идентификаторов __scl_bit, __sda_bit, __lcd_port подставит их значения. В случае модификации программы при использовании другого порта или битов для связи через шину I²C, а также другого порта для подключения LCD-дисплея, достаточно будет лишь заменить значения в указанных директивах .equ, не изменяя остального текста программы.

Директивами #include подключаются LCD-функции, функции температурного датчика LM75AD, микроконтроллера AT90S8515, ввода/вывода, задержки. Перед компиляцией препроцессор компилятора вставит вместо этих строк текст соответствующих файлов.

В строке **char lcd_buffer[33]**; объявляется глобальный символьный массив lcd_buffer, состоящий из 33 элементов, для хранения предназначенной для вывода на LCD-дисплей информации. Массив будет расположен в SRAM микроконтроллера. Все его элементы автоматически инициализируются со значением 0.

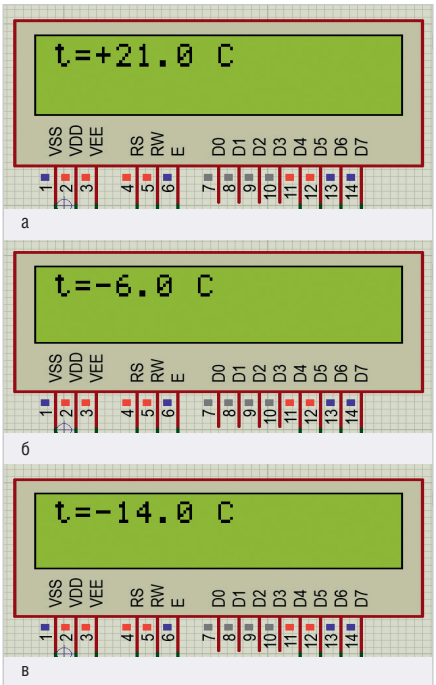

Рис. 14. Приближённый вид LCD-дисплея, на котором отображается температура, измеренная датчиком №: (а) 0, (б) 1, (в) 7

Далее выполняется основная функция программы void main(void), из которой осуществляется последовательный вызов:

- функции lcd_init(16) с параметром 16 (количество столбцов в LCD-модуле) для инициализации LCD-модуля, очистки дисплея и установки позиции для вывода символов в 0 ряд 0 столбца;
- функции i2c_init() для инициализации шины I²C;
- функции lm75_init(0,20,25,1) с параметрами 0 (адрес датчика), 20 (нижний предел температуры), 25 (верхний предел температуры), 1 (активным уровнем OS является высокий), которая инициализирует датчик температуры с адресом 0;
- функции lm75_init(1,–20,22,1) с параметрами 1 (адрес датчика), –20 (нижний предел температуры), 22 (верхний предел температуры), 1 (активным уровнем OS является высокий), которая инициализирует датчик температуры с адресом 1;
- функции lm75_init(7,–20,22,1) с параметрами 7 (адрес датчика), –20 (нижний предел температуры), 22 (верхний предел температуры), 1 (активным уровнем OS является высокий), которая инициализирует датчик температуры с адресом 7.

После этого программа переходит в бесконечный цикл отображения измеренной

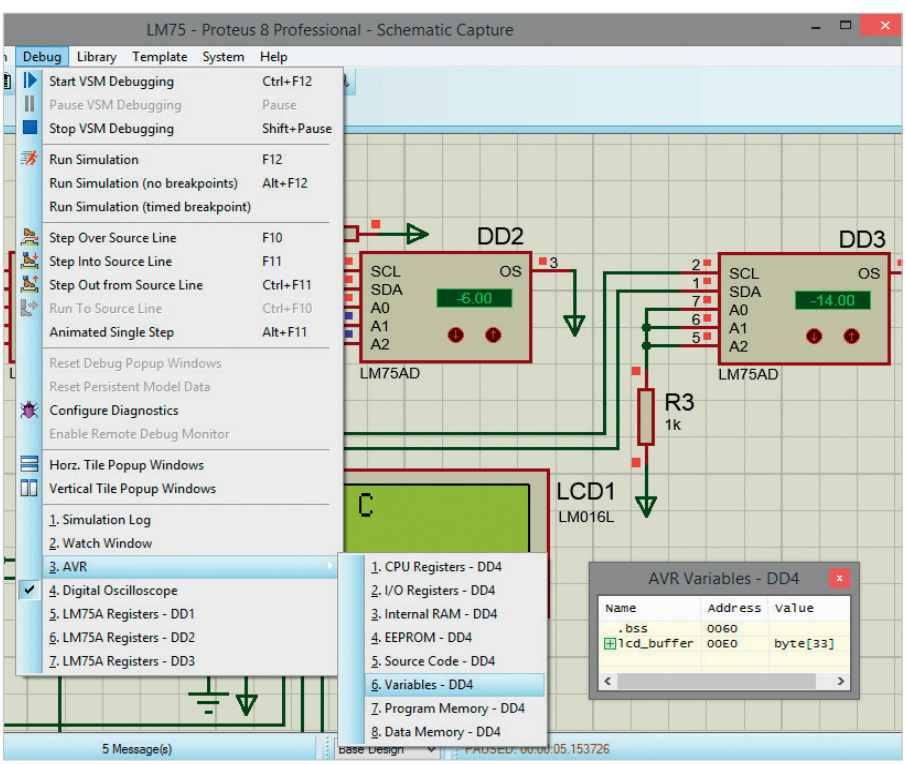

Рис. 15. Открытие окна AVR Variables

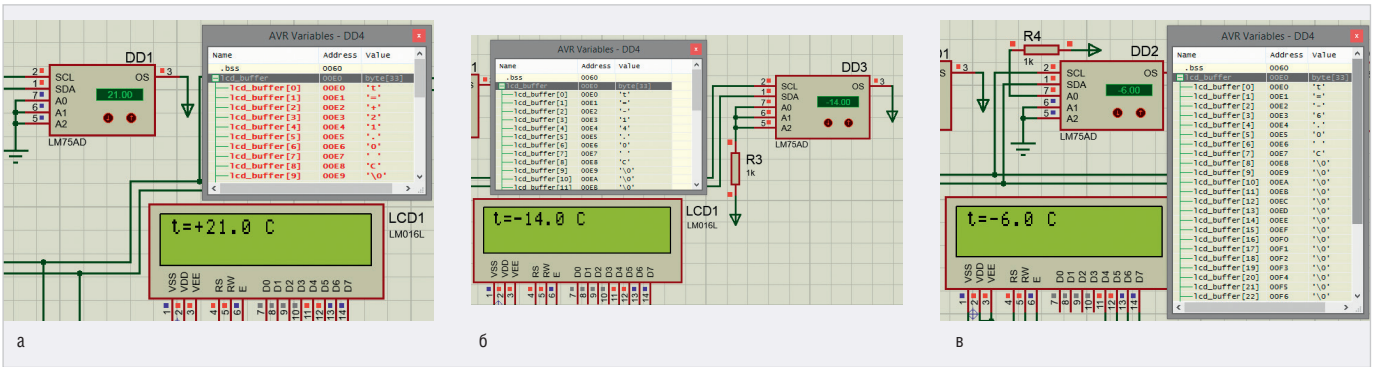

Рис. 16. Проверка значений массива lcd_buffer после отображения на экране дисплея измеренной датчиками LM75AD температуры: (а) +21 °С, (б) –14 °С, (в) –6 °С

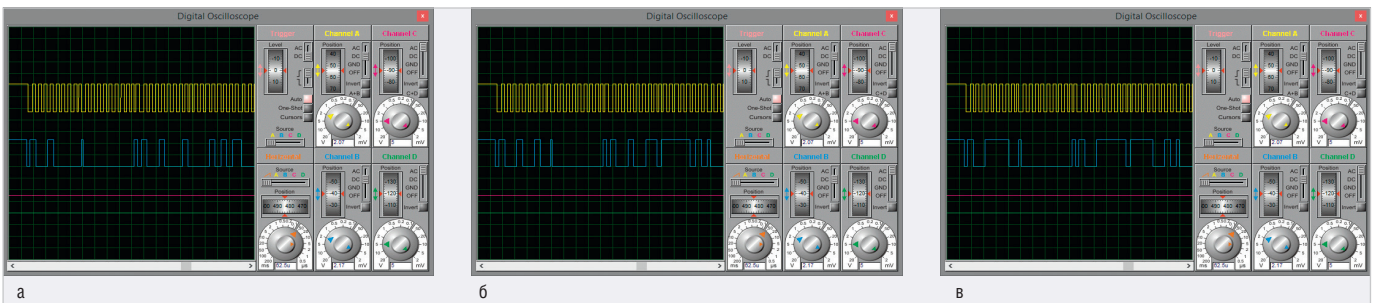

Рис. 17. Временны́е диаграммы сигналов SCL и SDA после передачи по интерфейсу I²C температуры, измеренной датчиком №: (а) 0, (б) 1, (в) 7

температуры while (1). При этом поочередно через каждые 2000 мс считывается и выводится на экран дисплея значение температуры, измеренное каждым датчиком (рис. 14а...14в). Дойдя до последнего датчика, измерение снова начинается с первого и т.д. Адреса датчиков хранятся в массиве **int i[3]={0,1,7}**. Для датчика с адресом 0 с помощью функции **lm75_temperature_10(0)** будет выполняться измерение температуры, запись полученного значения и дополнительного текста в массив lcd_buffer с помощью функции **sprintf(lcd_buffer,"t=%c%i.%u C", sign,temp/10,temp%10)** и вывод содержимого массива на экран дисплея с помощью функции **lcd_puts(lcd_buffer)**. Эти же действия будут выполнены для датчиков с адресом 1 и 7.

Вызов функции main происходит после подачи питания или аппаратного сброса на выводе RESET микроконтроллера. Затем последовательно выполняются функции инициализации дисплея (lcd_init), шины I²C (i2c_init) и датчика температуры LM75AD (lm75_init). После этого программа переходит к выполнению бесконечного цикла, в котором значение температуры считывается с датчика с помощью функции lm75_temperature_10. Далее выполняется запись полученной информации и дополнительного текста в массив lcd_buffer с помощью функции sprintf и его отображение с помощью функции lcd_puts на экране дисплея, предварительно очищенного с помощью функции lcd_clear. Рядом со значением отрицательной температуры выводится знак «–», если температура выше нуля, то знак «+». После этого происходит задержка в 2000 мс, и цикл повторяется.

Приостановим моделирование кнопкой Pause the simulation, or start up at time 0 if stopped и проверим значения элементов массива lcd_buffer. Для чего, используя команду основного меню Debug/AVR/Variables, откроем окно AVR Variables (рис. 15) и щелчком левой кнопки мыши по значку «+» раскроем список lcd_buffer. Как видно на рис. 16а, в ячейки 0...8 массива lcd_buffer записаны символы, которые совпадают с отображёнными на экране дисплея «t=+21.0 C»:

lcd_buffer[0] 't'
lcd_buffer[1] '='
lcd_buffer[2] '+'
lcd_buffer[3] '2'
lcd_buffer[4] '1'
lcd_buffer[5] '.'
lcd_buffer[6] '0'
lcd_buffer[7] ' '
lcd_buffer[8] 'C'.

После получения нового значения температуры, измеренной очередным датчиком, изменится значение символов в третьей и четвёртой ячейках массива lcd_buffer в окне AVR Variables (рис. 16б), а если температура ниже нуля, то и символ во второй ячейке массива. Если двузначное значение температуры сменится однозначным (рис. 16в), то символы, записанные в 5...8 ячейках, сдвинутся на место ячеек 4...7, т.е. на одну позицию влево в массиве.

На рис. 17а...17в показаны временны́е диаграммы сигналов SCL и SDA интерфейса I²C, полученные с помощью виртуального осциллографа. Отображение диаграмм на лицевой панели Digital Oscilloscope отрегулируем с помощью ручек управления. Установим в окнах Channel A, Channel B маленькую ручку в позицию 3 mV, большую ручку в позицию 2 V. Установим маленькую ручку в позицию 5 µs, а большую ручку в позицию 50 µs в окне Horizontal. Установим режим работы осциллографа: DC. Жёлтая диаграмма соответствует сигналу синхронизации SCL, который поступает на вход A осциллографа. Голубая диаграмма получена с входа B осциллографа и отображает данные, которые передаются через линию SDA.

Литература

1. ISIS Help, Labcenter Electronics, 2014.
2. CodeVisionAVR Help, HP InfoTech, 2014.
3. HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver). Hitachi, Ltd. 1998.
4. *Евстифеев А.В.* Микроконтроллеры AVR семейства Mega. Руководство пользователя. М.: Издательский дом «Додэка-XXI», 2007.
5. Proteus VSM Help, Labcenter Electronics, 2020.
6. *Хартов В.Я.* Микроконтроллеры AVR. Практикум для начинающих. М.: Издательство МГТУ им. Н.Э. Баумана, 2007.