Работа с универсальным синхронно/асинхронным приёмопередатчиком USART в программной среде Proteus 8.11

Татьяна Колесникова (beluikluk@gmail.com)

В статье рассматривается проектирование схем микроэлектронных устройств с использованием модуля USART в Proteus. Приведены примеры моделирования схем, в которых проводится обмен данными через последовательный интерфейс между виртуальным терминалом, алфавитно-цифровым дисплеем и микроконтроллерами AVR (семейства Mega) и STM32 (семейства Cortex-M3) под управлением программы, написанной на языке С или ассемблере, с применением одного или сразу двух модулей USART. В ходе выполнения программы отслеживается состояние регистров управления модулем USART. С помощью осциллографа и логического анализатора осуществлён контроль входных/выходных сигналов, присутствующих в цепях исследуемых схем.

Введение

USART (Universal Synchronous Asynchronous Receiver Transmitter) – это модуль последовательного ввода-вывода, который используется для обмена данными между цифровыми устройствами (например, персональным компьютером и его периферией). Модуль подходит для организации связи между двумя микроконтроллерами, а также для связи любых устройств, где данный протокол поддерживается. На базе USART построены многие промышленные интерфейсы передачи данных: RS-232 (COM-порт), RS-422, RS-485, «токовая петля», которые отличаются физической средой передачи логической 1 и 0 при одинаковой структуре кадра и временны́х параметров. USART может работать в трёх режимах:

- асинхронный, полный дуплексный режим;
- ведущий синхронный, полудуплексный режим;
- ведомый синхронный, полудуплексный режим.

Модуль приёмопередатчика обеспечивает полнодуплексный обмен по последовательному каналу, при этом скорость передачи данных может варьироваться в довольно широких пределах. Длина посылки может составлять от 5 до 9 бит. В модуле присутствует схема контроля и формирования бита чётности. Модуль USART обнаруживает следующие внештатные ситуации:

- переполнение;
- ошибка кадрирования;
- неверный старт-бит.

Для уменьшения вероятности сбоев в модуле реализована функция фильтрации помех. Для взаимодействия с программой в микроконтроллере, как правило, предусмотрены прерывания, запрос на генерацию которых формируется при наступлении следующих событий:

- «передача завершена»;
- «регистр данных передатчика пуст»;
- «приём завершён».

Интерфейс USART задействует 3 линии ввода-вывода:

- TxD – передача данных;
- RxD – приём данных;
- XCK – тактовый сигнал (используется только в синхронном режиме).

Рассмотрим работу с USART на примере микроконтроллеров AVR (микросхемы ATmega128 и ATmega16) и STM32 (микросхема STM32F103C4), для чего воспользуемся программой компьютерного моделирования электронных схем Proteus (см. рис. 1). Основное отличие программы Proteus от аналогичных по назначению пакетов программ, например, Multisim, заключается в развитой системе симуляции (интерактивной пошаговой отладки и отладки в режиме реального времени) для различных семейств микроконтроллеров, среди которых Mega и Cortex-M3.

Первый этап проектирования узла печатной платы в системе Proteus [1] – это разработка схемы электрической принципиальной в редакторе Schematic Capture. На этой стадии проектирования выбирают компоненты из библиотеки, размещают их в рабочем поле чертежа, выполняют связь компонентов при помощи цепей и шин. При необходимости модифицируют свойства компонентов, добавляют текстовые надписи.

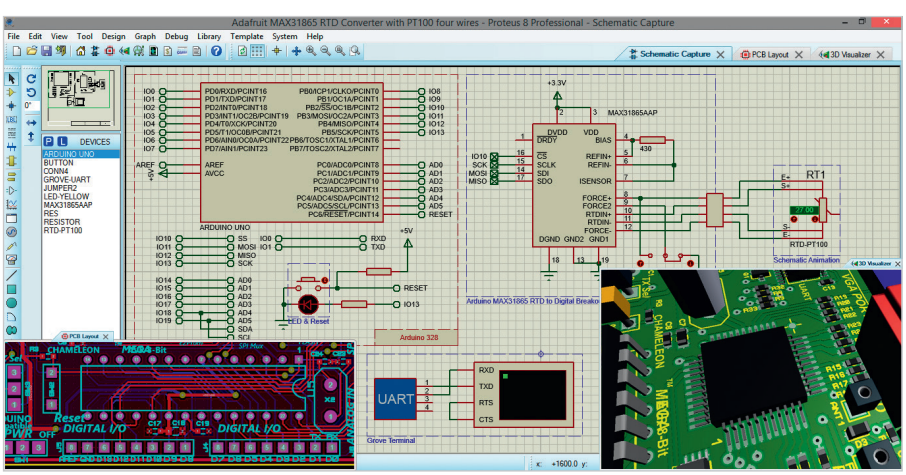

Рис. 1. Программная среда Proteus 8.11

Приём и передача данных через последовательный интерфейс USART в микроконтроллерах Cortex-M3 в Proteus

Работа с универсальным синхронно/асинхронным приёмопередатчиком USART

Микроконтроллеры семейства Cortex-M3 имеют в своём составе до трёх модулей универсального синхронно/асинхронного приёмопередатчика USART. В микросхеме STM32F103C4 имеется три таких модуля USART (USART1,

USART2, USART3), а также 2 модуля UART (UART4, UART5), которые предоставляют практически те же возможности, что и USART, за исключением функций синхронного обмена данными и аппаратного контроля передачи (соответственно, для этих модулей недоступен режим Smartcard). Кроме того, для модуля UART5 недоступна работа через DMA. Все модули приёмопередатчиков обеспечивают полнодуплексный обмен по последовательному каналу. Длина посылки может составлять от 8 до 9 битов. Во всех модулях в обязательном порядке присутствуют схемы контроля и формирования бита чётности. Выводы микроконтроллера, используемые модулями USART, являются линиями ввода/вывода общего назначения (микросхема STM32F103C4 имеет два 16-разрядных порта ввода/вывода с возможностью управления их битовыми линиями). Назначение выводам функции USART осуществляется в регистрах настройки GPIO. В микроконтроллере STM32F103C4 модулем USART1 используются линии PA10 (RXD) – вход USART1, PA9 (TXD) – выход USART1, PA8 (XCK) – вход/выход внешнего тактового сигнала USART1. Однако, используя регистры AFIO, вход и выход USART1 можно перенести на выводы PB7 и PB6 соответственно. USART2 для приёма/передачи данных использует выводы PA3 и PA2.

Модуль USART состоит из трёх основных частей: блока тактирования, блока передатчика и блока приёмника. Буферные регистры приёмника и передатчика располагаются по одному адресу пространства ввода/вывода. Для доступа к ним используют младшие 9 бит регистра USART_DR. При чтении регистра USART_DR выполняется обращение к буферному регистру приёмника RDR, при записи – к буферному регистру передатчика TDR. Прямого доступа к регистрам TDR и RDR нет.

Для управления модулем USART используются регистры USART_SR, USART_BRR, USART_CR1, USART_CR2, USART_CR3. Работа модуля USART разрешается установкой в лог. «1» бита UE (USART Enable) регистра USART_CR1. Работа передатчика разрешается установкой в лог. «1» бита TE (Transmitter Enable) регистра USART_CR1. При установке бита вывод TXD подключается к передатчику USART и начинает функционировать как выход, независимо от установок регистров управления портом. Если используется синхронный режим работы, то переопределяется также функционирование вывода XCK. Передача инициируется записью передаваемых данных в буферный регистр передатчика, т.е. в регистр данных USART_DR. После этого данные пересылаются из регистра USART_DR в сдвиговый регистр передатчика. После пересылки слова данных в сдвиговый регистр флаг TXE регистра USART_SR устанавливается в 1, что означает готовность передатчика к получению нового слова данных. В этом состоянии флаг остаётся до следующей записи в буфер.

Выключение передатчика осуществляется сбросом бита TE регистра USART_CR1. Если в момент выполнения этой команды осуществлялась передача, сброс бита произойдёт только после завершения текущей и отложенной передач, то есть после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD может использоваться как контакт ввода/вывода общего назначения. Настройку скорости передачи данных выполняют записью соответствующего значения в регистр USART_BRR.

Работа приёмника разрешается установкой в лог. «1» бита RE регистра USART_CR1. При установке бита вывод RXD подключается к приёмнику USART и начинает функционировать как вход, независимо от установок регистров управления портом. Если используется синхронный режим работы, переопределяется также функционирование вывода XCK. Выключение приёмника осуществляется сбросом бита RE регистра USART_CR1. В отличие от передатчика, приёмник выключается сразу же после сброса бита, а значит, кадр, принимаемый в этот момент, теряется. Кроме того, при выключении приёмника очищается его буфер, то есть теряются также все непрочитанные данные. При выключенном приёмнике вывод RXD может использоваться как контакт ввода/вывода общего назначения.

Включение контроля чётности выполняют установкой в лог. «1» бита PCE регистра USART_CR2. В этом же регистре задают количество стоповых битов, определив значение флага STOP (00-1, 01-0,5, 10-2, 11-1,5 стоп- бита). Управляя значениями битов регистров модуля USART микроконтроллеров Cortex-M3, реализовывают процесс обмена данными. Для этого в регистре настроек USART_CR1 используют флаги:

- UE – 13-й бит регистра, установка 1 в котором включает модуль USART, 0 – выключает;
- M – 12 бит регистра, в котором определяют количество информационных битов в посылке (0–8 бит, 1–9 бит);
- PCE – 10-й бит регистра, установка единицы в котором включает контроль чётности, нуля – выключает;
- PS – 9-й бит регистра используют для проверки нечётности (1) или чётности (0);
- TE – 3-й бит регистра, где определяют состояние передатчика: 1 – включён, 0 – отключён;
- RE – 2-й бит регистра, где определяют состояние приёмника: 1 – включён, 0 – отключён;
- RWU – 1-й бит управляет режимом работы приёмника USART, когда в нём записана единица – приёмник переключается в режим ожидания, когда ноль – устанавливается активный режим.

В регистре состояния USART_SR используют флаги:

- TXE – 7-й бит регистра USART_SR устанавливается в 1 после передачи всех битов регистра TDR в сдвиговый регистр передатчика, что указывает на то, что буферный регистр пуст и готов к приёму новых данных. Флаг сбрасывается автоматически после записи байта данных в регистр USART_DR;
- TC – 6-й бит регистра устанавливается в 1 после передачи всех битов из сдвигового регистра передатчика, если при этом установлен бит TXE (то есть если в регистре данных передатчика нет новых данных для передачи). Сбрасывается автоматически выполнением программной последовательности: чтение регистра USART_SR с последующей записью в регистр USART_DR;
- RXNE – 5-й бит регистра, устанавливается в 1 после перемещения принятых данных из сдвигового регистра приёмника в регистр данных приёмника, что указывает на то, что буферный регистр приёма не пуст. Данные должны быть считаны до прихода следующих, иначе они будут потеряны. Флаг сбрасывается автоматически при чтении регистра USART_DR;
- ORE – 3-й бит регистра устанавливается в 1, если в сдвиговом регистре приёмника готова очередная порция данных для передачи в регистр данных приёмника, но при этом из регистра данных приёмника ещё не считали предыдущие принятые данные. При возникновении этой ошибки данные в регистре RDR сохраняют-

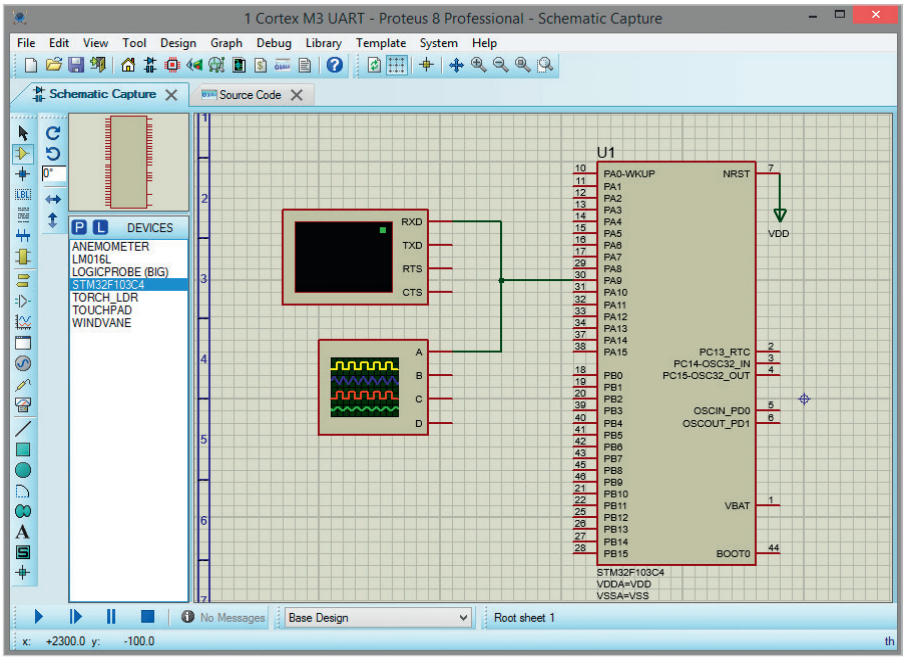

Рис. 2. Новый проект на базе микроконтроллера STM32F103C4 в Proteus 8.11

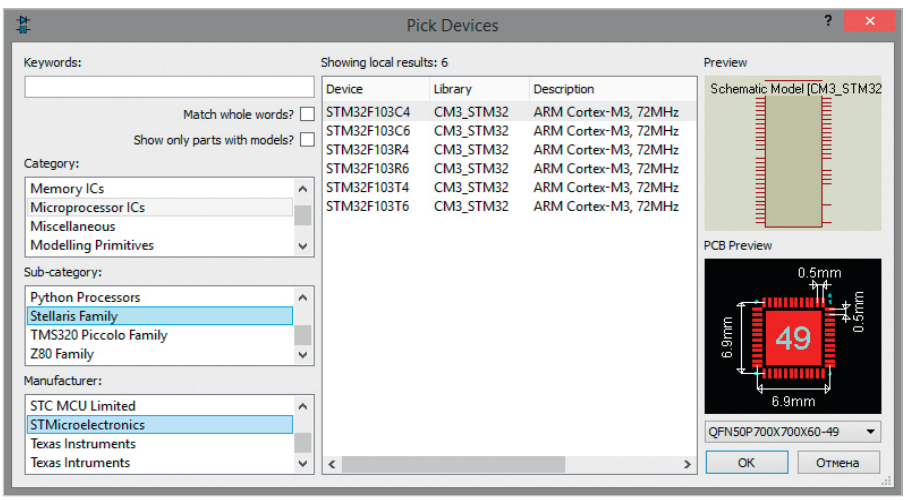

Рис. 3. Выбор микросхемы STM32F103C4 из библиотеки микроконтроллеров Microprocessor ICs базы данных компонентов Proteus

ся, а в сдвиговом регистре приёмника перезаписываются;

- PE – 0-й бит регистра устанавливается в единицу при обнаружении ошибки чётности.

Передача данных через последовательный интерфейс USART

Рассмотрим работу модуля USART на конкретном примере. Передадим программным способом на экран виртуального терминала комбинацию символов английского алфавита «ABC» через последовательный интерфейс. Для этого создадим в Proteus новый проект с использованием микроконтроллера STM32F103C4 (см. рис. 2), микросхема которого находится в разделе Stellaris Family библиотеки Microprocessor ICs базы данных компонентов Proteus (см. рис. 3).

Щёлкнув левой кнопкой мыши на панели INSTRUMENTS (рис. 4) строку с названием VIRTUAL TERMINAL, а затем OSCILLOSCOPE разместим мышью в рабочем поле проекта виртуальный терминал и виртуальный осциллограф, которым воспользуемся для просмотра осциллограммы работы USART. Чтобы открыть панель INSTRUMENTS на левой панели схемного редактора, нажимают пиктограмму Virtual Instruments Mode. Подсоединим вывод PA9 (TXD) микроконтроллера к выводу RXD виртуального терминала, а также к каналу A осциллографа.

В окне настроек микроконтроллера Edit Component в поле Crystal Frequency установим частоту его работы 2 МГц (см. рис. 5а). В окне настроек терминала (см. рис. 5б) определим значения следующих параметров:

- Baud Rate – скорость обмена данными (9600 бод);
- Data Bits – формат пакета данных (8 бит);
- Parity – контроль чётности (отсутствует – NONE);
- Stop Bits – количество стоповых битов (1).

Окна настроек открывают двойным щелчком левой кнопкой мыши по размещённому на схеме компоненту.

Для проверки работы собранной схемы на языке программирования С была написана программа, код которой приведён ниже:

```
#include <stm32f1xx.h> // подключение библиотеки функций
#define F_CPU 2000000 // рабочая частота микроконтроллера
#define baudrate 9600L // скорость обмена данными

int init_USART() // инициализация USART и GPIO
{ // включаем тактирование USART1
RCC->APB2ENR |= RCC_APB2ENR_USART1EN;
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

// настройка линии PA9 (TXD) порта PA
// биты CNF = 10, биты MODE = 01
GPIOA->CRH = 0x00000090;

// конфигурация USART1
USART1->CR1 = (1<<13); // разрешаем USART1, сбрасываем остальные биты
USART1->BRR = (F_CPU/ (16 * baudrate))*16; // рассчитываем значение для регистра BRR
USART1->CR1 |= (1<<3); // включаем передатчик
USART1->CR2 = 0; // сбрасываем все флаги регистров CR2 и CR3
USART1->CR3 = 0;
}

// вывод данных
void send_USART (char value) {
while ( USART1->SR == ((0<<6)|(0<<7)) ) { } // ожидаем, когда очистится буфер передачи
USART1->DR = value; // помещаем данные в буфер, начинаем передачу
}
```

```
  int main(void) // начало программы
  {
  init_USART(); // инициализация USART и GPIO
  send_USART('A'); // вывод на экран символа 'A'
  send_USART('B'); // вывод на экран символа 'B'
  send_USART('C'); // вывод на экран символа 'C'
  }
```

После того как в рабочей области проекта собрана схема, а на вкладке Source Code введён код программы, можно запускать моделирование, для чего предусмотрена кнопка Run the simulation в левом нижнем углу окна программы. Как видно из рис. 6, разработанный проект функционирует верно: на экран виртуального терминала была выведена указанная в коде программы комбинация символов. На этом же рисунке показана осциллограмма работы USART.

Для подключения виртуального осциллографа к схеме используется его пиктограмма. Для настройки прибора и наблюдения формы исследуемого сигнала предназначена лицевая панель, которая открывается после запуска симуляции схемы. В её левой части расположен графический дисплей, который предназначен для графического отображения формы сигнала (напряжения по вертикальной оси и времени по горизонтальной оси). Панель управления осциллографа находится в правой части его лицевой панели и предназначена для настройки отображения измеряемого сигнала. Результаты работы четырёхканального осциллографа отображаются на экране графического дисплея в виде четырёх кривых, которые представляют четыре сигнала со входов A, B, C, D. Для получения осциллограммы работы модуля USART настроим параметры осциллографа так, как показано на рис. 6.

Так как в микроконтроллере STM32F103C4 имеется 3 модуля USART, при написании программного кода необходимо указывать номер модуля, к которому мы обращаемся. В нашем примере это USART1. Программа инициализации микроконтроллера содержит три функции: int init_ USART(), void send_USART(char value), int main(void). В функции int init_ USART() выполняется включение тактирования модуля USART1 и порта PA (команды RCC->APB2ENR |= RCC_APB2ENR_USART1EN и RCC->APB2ENR |= RCC_APB2ENR_IOPAEN), настройка режима работы вывода PA9 (команда GPIOA->CRH = 0x00000090), включение модуля USART1 и передатчика (команды USART1->CR1 = (1<<13) и USART1->CR1 |= (1<<3)). В функции void send_USART(char value) выполняется проверка 6 и 7 бита регистра USART1_SR (команда while (USART1->SR = = ((0<<6)|(0<<7))) { }) – как только его флаги TXE и TC установятся в 1, начнётся передача данных (команда USART1->DR = value). Из функции int main(void) выполняется вызов функций инициализации и вывода данных.

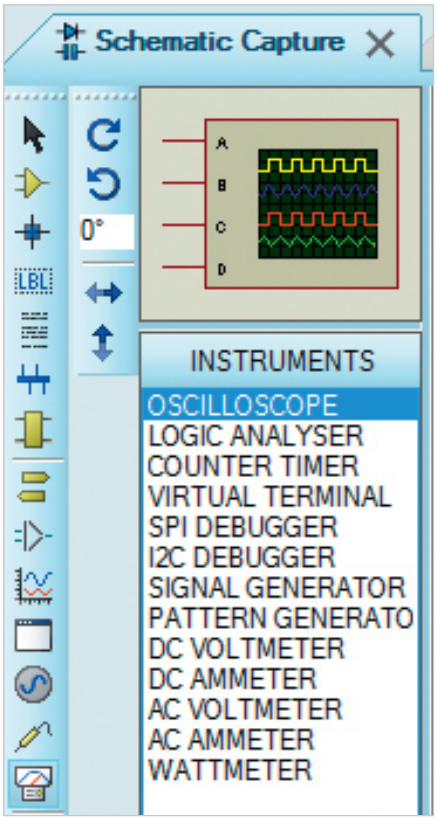

Рис. 4. Открытие панели INSTRUMENTS с помощью пиктограммы Virtual Instruments Mode

В микроконтроллере STM32F103C4 линии ввода/вывода, в зависимости от настроек, могут быть общего назначения (GPIO) либо использоваться как специальные. Настройка портов GPIO для цифрового ввода/вывода требует выпол-

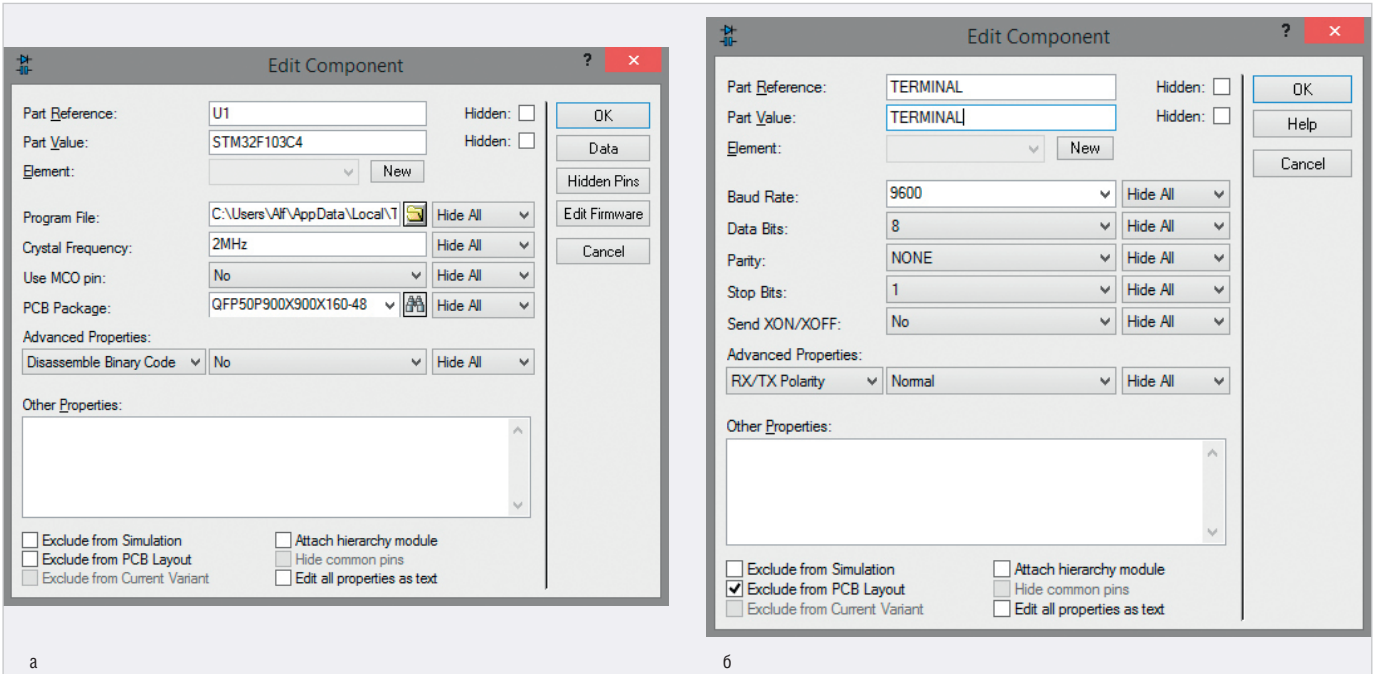

Рис. 5. Настройка параметров: (а) Crystal Frequency микроконтроллера STM32F103C4 и (б) Baud Rate, Data Bits, Parity, Stop Bits виртуального терминала

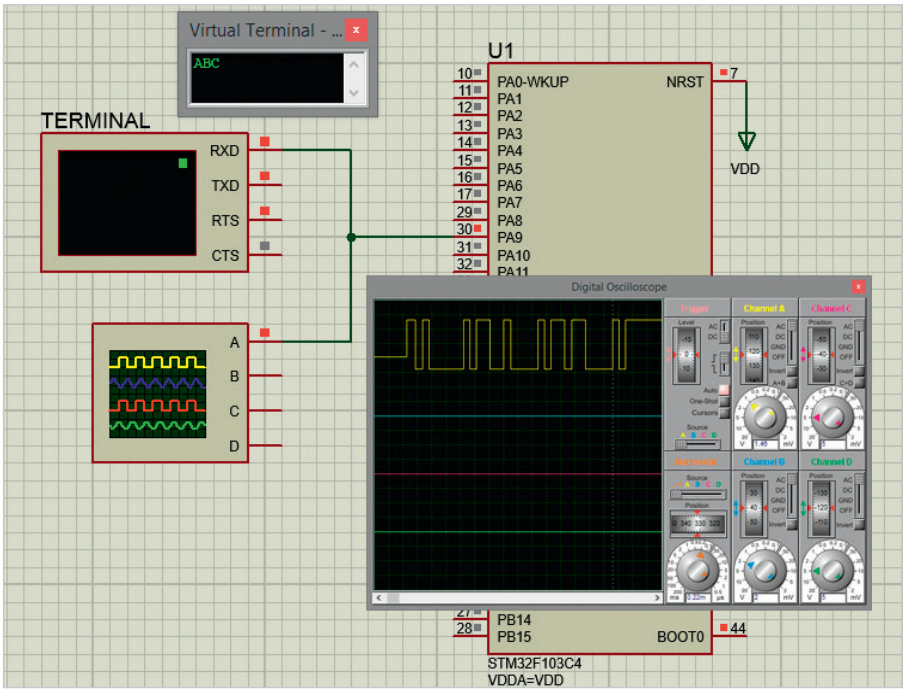

Рис. 6. Вывод символов ABC на экран виртуального терминала через последовательный интерфейс USART микроконтроллера STM32F103C4

нения двух основных действий: подключения тактового сигнала и определения режима работы. В рассмотренном выше примере подключение тактового сигнала реализовано с помощью следующего фрагмента кода: RCC->APB2ENR |= RCC_APB2ENR_IOPAEN, которым даётся указание микроконтроллеру подсоединить линии порта PA к шине APB2, установив соответствующий бит IOPAEN (где PA – имя порта) в регистре разрешения тактирования периферийных блоков RCC_APB2ENR. Запись RCC->APB2ENR представляет собой обращение к регистру APB2ENR из группы регистров тактирования и контроля RCC. Аналогичным образом с помощью команды RCC->APB2ENR |= RCC_APB2ENR_USART1EN включается тактирование модуля USART1.

Для переключения режимов работы портов ввода/вывода STM32 Cortex-M3 используются два 32-разрядных регистра для каждого GPIO. Они позволяют произвольно настроить режим работы любой отдельной линии. Регистр GPIOx_CRL отвечает за линии с номерами от 0 до 7, GPIOx_CRH – за линии от 8 до 15 (где x – это имя порта). В нашем примере для последовательной передачи данных на экран виртуального терминала необходимо настроить режим работы линии PA9, что осуществляется с помощью 1 бита регистра конфигурации GPIOA_CRH. Для линии PA9 в регистре GPIOA_CRH имеется два двухразрядных поля: CNF1 и MODE1. Первое определяет тип работы линии, второе – направление передачи информации по линии (все биты доступны для чтения/записи).

Запись GPIOA->CRH = 0x00000090 в коде программы означает, что линия PA9 порта PA микроконтроллера STM32F103C4 имеет специальное назначение (работает на вывод данных по USART), для чего для этой линии в поле MODE записано значение 01 (линия работает на вывод данных с максимальной частотой переключения 10 МГц), а в поле CNF – значение 10 (цифровой выход с альтернативной функцией). Двоичный код 1001 соответствует шестнадцатеричному значению – 9, которое записано в первый разряд регистра конфигурации GPIOA_CRH линий 8...15 порта PA.

Каждый разряд регистров GPIOx_CRL и GPIOx_CRH управляет своим разрядом порта. Если в каком-либо разряде регистров GPIOx_CRL и GPIOx_CRH в поле MODE записана комбинация 00, то соответствующий разряд порта работает как вход. Если же в поле MODE этого разряда записаны значения 01, 10, 11, то разряд порта работает как выход общего назначения (если в поле CNF записаны значения 00 или 01) или специального (если в поле CNF записаны значения 10 или 11).

Включение модуля USART1 и его передатчика выполняется установкой флагов UE и TE регистра USART1_CR1, то есть записью логической единицы в 13 и 3 биты регистра USART1_CR1 с помощью команд USART1->CR1 = (1<<13) и USART1->CR1 |= (1<<3). Флаги регистров USART1_CR2 и USART1_CR3 в нашем примере не имеют значения. Расчёт значения скорости передачи данных и его запись в регистр USART1_BRR выполняется с помощью команды USART1->BRR = (F_CPU/ (16 * baudrate))*16.

Таким образом, для того чтобы послать данные через USART в микроконтроллере STM32F103C4, необходимо:

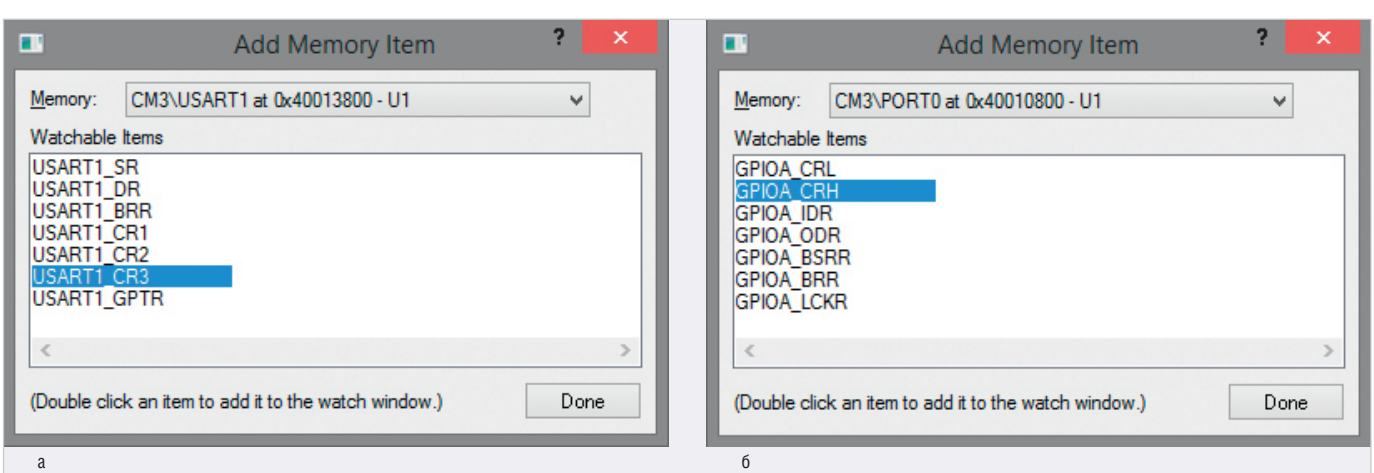

Рис. 7. Добавление в окно Watch Window регистров: (а) модуля USART1, (б) порта PA

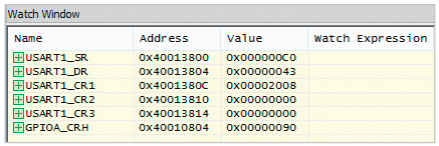

Рис. 8. Список регистров в окне Watch Window

- включить тактирование выбранного модуля USARTx (где x – номер модуля) и порта ввода/вывода, через который будет вестись передача данных;
- настроить режим работы линии передачи на вывод данных с альтернативной функцией, записав в соответствующий разряд GPIOx_CRH нужную комбинацию бит;
- разрешить работу с выбранным модулем USARTx и включить передатчик;
- записать в регистр USARTx_BRR значение скорости передачи;
- после установки в 1 флагов TXE и TC регистра USARTx_SR записать данные в регистр USART1_DR.

После отображения на экране виртуального терминала комбинации символов «ABC» приостановим симуляцию кнопкой Pause the simulation, or start up at time 0 if stopped (кнопка находится в левом нижнем углу окна программы) и проверим содержимое регистров микроконтроллера. Для этого, используя команду основного меню Debug / Watch Window, откроем окно Watch Window, где можно размещать регистры микроконтроллера и отслеживать их содержимое в ходе выполнения программы. Добавление регистра выполняется щелчком правой кнопки мыши в области окна Watch Window и выбором в открывшемся контекстном меню пункта Add Items (By Name) – добавить элементы по имени. В результате чего будет открыто окно Add Memory Item, которое содержит список групп всех регистров микроконтроллера. В нашем примере представляют интерес регистры модуля USART1 (USART1_CR1, USART1_CR2, USART1_CR3, USART1_SR, USART1_DR) и порта PA (GPIOA_CRH), с которыми ведётся работа в тексте программы инициализации микроконтроллера. Программа переключает биты регистров, осуществляя управление параметрами микроконтроллера. Для добавления регистров USART (см. рис. 7а) в окно Watch Window из выпадающего списка в поле Memory окна Add Memory Item выберем пункт CM3\USART1 at 0x40013800-U1 (имена регистров появятся в поле Watchable Items), двойным щелчком левой кнопки мыши выберем нужные элементы и нажмём кнопку Done.

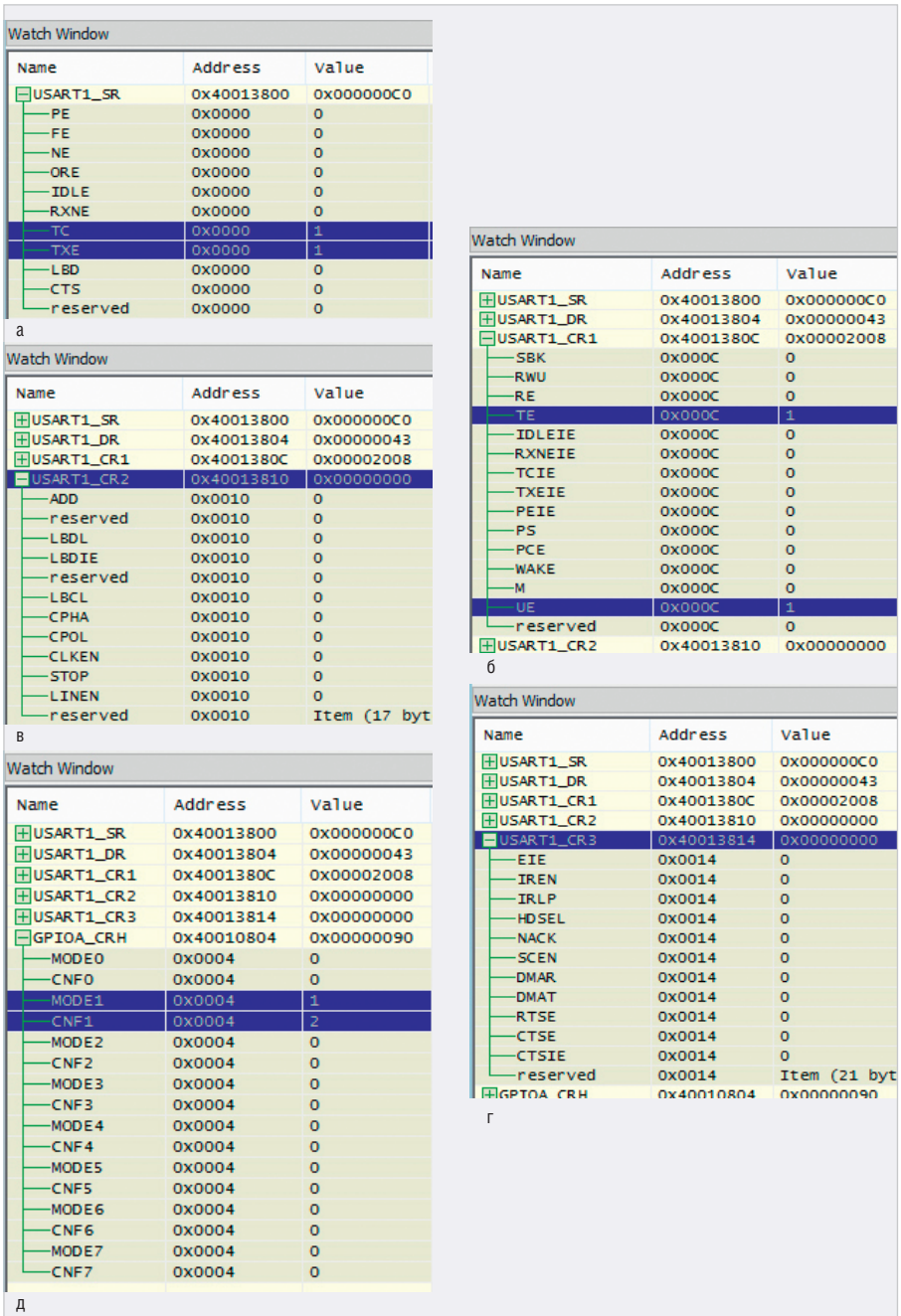

Рис. 9. Значения битов регистров: (а) USART1_SR, (б) USART1_CR1, (в) USART1_CR2, (г) USART1_CR3 модуля USART1 и (д) GPIOA_CRH порта PA после выполнения кода программы инициализации микроконтроллера

Для добавления регистров порта PA (см. рис. 7б) в окно Watch Window в поле Memory окна Add Memory Item выберем пункт CM3\PORT0 at 0x40010800-U1. Добавить элементы в окно Watch Window также можно и по адресу, для чего в окне вызывают контекстное меню и выбирают в нём пункт Add Items (By Address).

Элементы в окне Watch Window располагаются в виде списка (см. рис. 8), который раскрывают щелчком левой кнопки мыши по значку «+», при этом становятся доступными для просмотра адрес и значения функционально связанных битов регистра. В биты TC и TXE регистра USART1_SR (см. рис. 9а) и в биты TE и UE регистра USART1_CR1 (см. рис. 9б) записаны логические единицы, в биты регистров USART1_CR2 (см. рис. 9в), USART1_CR3 (см. рис. 9г) записаны логические нули, в биты MODE1 и CNF1 регистра GPIOA_CRH (см. рис. 9д) записаны шестнадцатеричные значения 1 (двоичное 01) и 2 (двоичное 10), что соответствует логике работы программы инициализации микроконтроллера.

Приём данных через последовательный интерфейс USART

Рассмотрим пример, в котором программным способом выполняется чте-

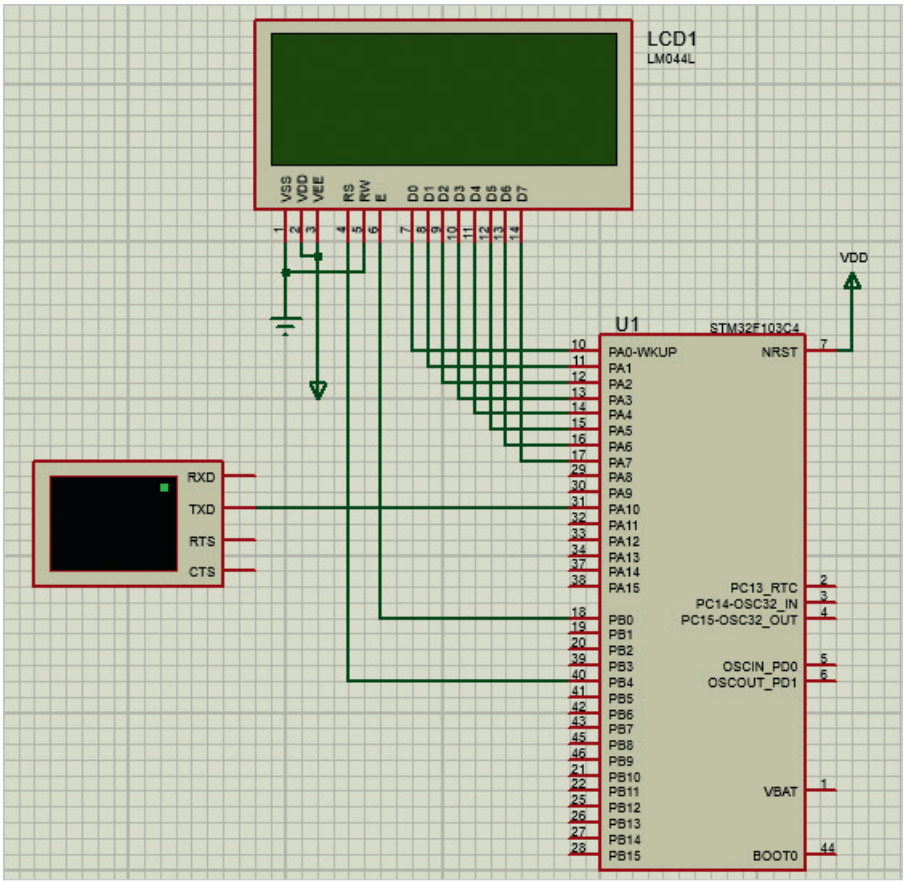

Рис. 10. Система ввода/вывода текстовых данных на базе микроконтроллера STM32F103C4 в Proteus 8.11

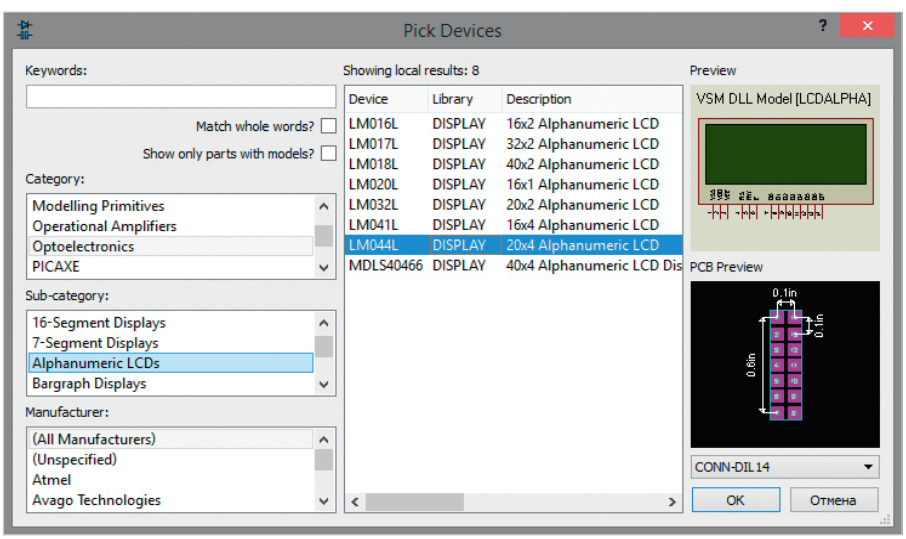

Рис. 11. Добавление в проект Schematic Capture микросхемы алфавитно-цифрового дисплея LM044L

ние текстовой информации с экрана виртуального терминала через последовательный интерфейс USART микроконтроллера и её вывод на экран алфавитно-цифрового дисплея, в качестве которого применим микросхему LM044L. Ввод текстовой информации осуществляется с помощью клавиатуры компьютера. Управление электронной системой ввода/вывода организуем с помощью микроконтроллера STM32F103C4. Передачу данных между периферийными устройствами и микроконтроллером обеспечим с помощью модуля USART (чтение данных с экрана виртуального терминала) и 8-разрядной шины данных/команд LCD-дисплея (вывод данных на экран дисплея). Интерфейс обмена данными настроим программно с помощью управляющих команд микроконтроллера. Связь между устройствами осуществляется с помощью линий ввода/вывода общего назначения микроконтроллера. Для вывода данных на экран дисплея и передачи управляющих команд воспользуемся линиями PA0...PA7, а для чтения данных с экрана терминала – линией PA10 порта PA микроконтроллера STM32F103C4, которая настроена как специальная. Чтение и вывод данных выполняется посимвольно.

Создадим в Proteus новый проект с использованием микроконтроллера STM32F103C4. Добавим в проект виртуальный терминал и подсоединим его вывод TXD к выводу PA10 (RXD) микроконтроллера. Добавим в рабочее поле проекта микросхему алфавитно-цифрового дисплея LM044L (см. рис. 10), которая находится в разделе Alphanumeric LCDs библиотеки Optoelectronics (см. рис. 11). Микросхема LM044L имеет 14 контактов, назначение которых следующее:

- V_{ss} – GND;
- V_{dd} – напряжение питания +5 В;
- V_{ee} – 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).

В представленном примере вывод данных на экран дисплея разрешением 20 символов на 4 строки выполнен в 8-разрядном режиме [3]. Подача управляющих сигналов через подключённые к портам микроконтроллера STM32F103C4 линии выполняется программно. 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 PA0...PA7, а выводы RS и E к выводам PB4 и PB0 порта микроконтроллера так, как показано на рис. 10. 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

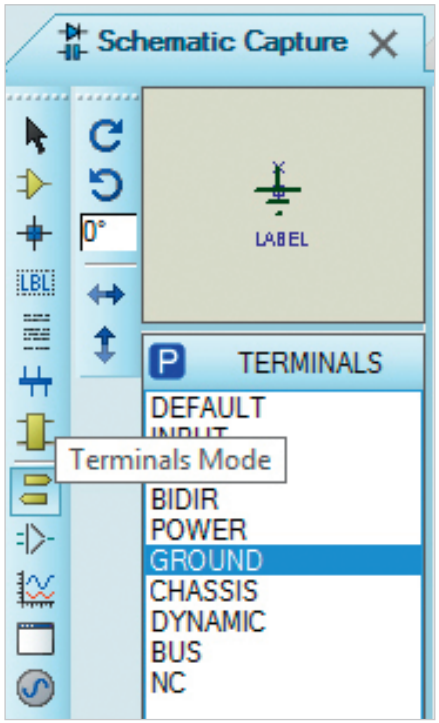

Рис. 12. Выбор символа «земли» на панели TERMINALS

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 (см. рис. 12) 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 в поле Advanced Properties из выпадающего списка выбирают пункт Clock Frequency (тактовая частота), значение которой в нашем примере составляет 250 кГц (см. рис. 13а). В окне свойств микроконтроллера указывают путь к файлу прошивки на диске компьютера (поле Program File) и значение частоты (поле Crystal Frequency) – в нашем примере 2 МГц (см. рис. 13б). Другие параметры оставляют без изменений. Окна свойств открывают двойным щелчком левой кнопки мыши по выделенному на схеме компоненту.

В окне настроек терминала определим значения следующих параметров:

- Baud Rate – скорость обмена данными (9600 бод);
- Data Bits – формат пакета данных (8 бит);
- Parity – контроль чётности (отсутствует – NONE);
- Stop Bits – количество стоповых битов (1).

Для графического отображения сигналов воспользуемся виртуальным логическим анализатором, добавить который в рабочую область проекта можно посредством выбора пункта LOGIC ANALYSER на панели INSTRUMENTS и щелчка левой кнопкой мыши в области схемы (см. рис. 14). Панель INSTRUMENTS открывают нажатием пиктограммы Virtual Instruments Mode на левой панели схемного редактора. Подсоединим выводы A0...A7 логического анализатора к линиям D0...D7 микросхемы LM044L, а выводы A9, A10 к линиям RS, E соответственно (см. рис. 14). После запуска моделирования схемы прибор снимает входные значения со своих выводов и отображает полученные данные в виде прямоугольных импульсов на часовой диаграмме во временно́й области лицевой панели.

Логический анализатор оперирует последовательно записанными в буфер захвата входными цифровыми данными. Процесс захвата данных запускается при помощи кнопки Capture окна Trigger лицевой панели прибора. Спустя некоторое время после выполнения условий переключения этот процесс останавливается, а кнопка меняет свой цвет при записи и после её завершения. Результат – содержимое буфера захвата – отображается на дисплее. В окне Horizontal расположены две ручки: Display Scale и Capture Resolution. При помощи первой производится масштабирование отображения диаграммы, при помощи второй – подстройка разрешения.

Видимость вводимого текста на экране виртуального терминала задают командой Echo Typed Characters контекстного меню (см. рис. 15), которое вызывают после запуска симуляции схемы щелчком правой кнопки мыши в области открывшегося окна терминала.

Необходимо учитывать, что большинство операций, выполняемых контроллером управления дисплеем (в нашем примере это HD44780 [3]), занимают значительное время, около 40 мкс, а время выполнения некоторых доходит до единиц миллисекунд. Поэтому в программе управления жидкокристаллическим модулем совершению

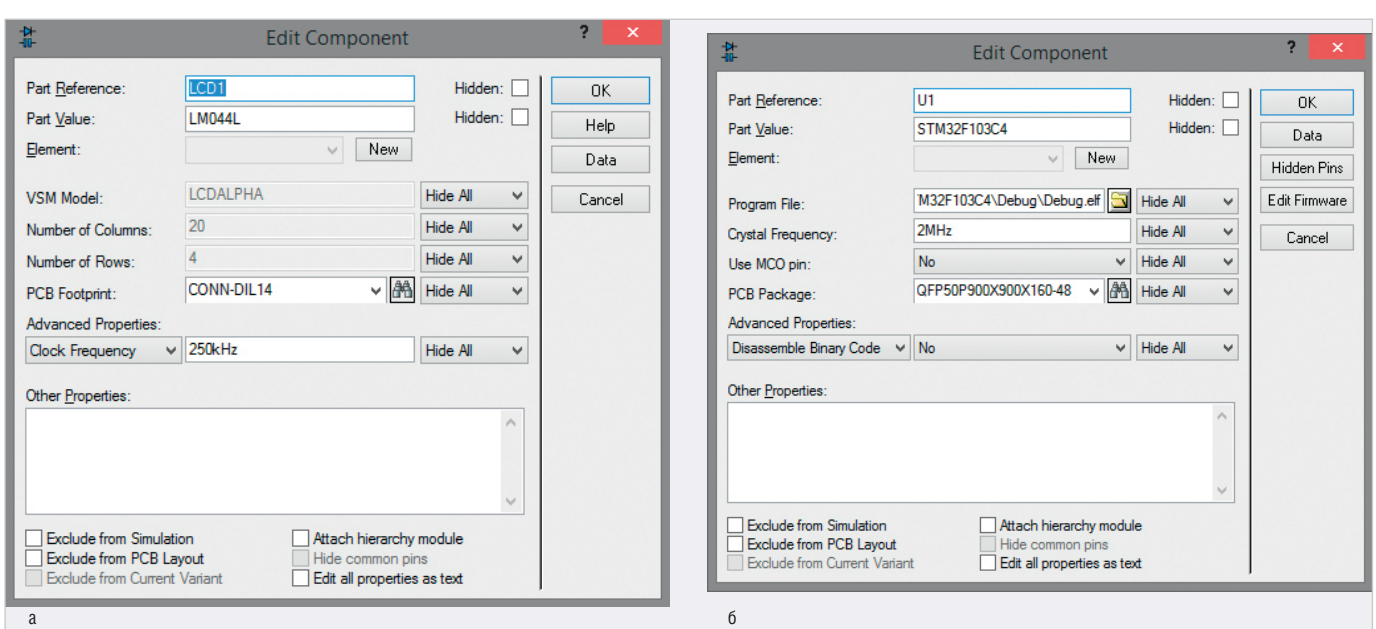

Рис. 13. Окно свойств: (а) дисплея LM044L, (б) микроконтроллера STM32F103C4

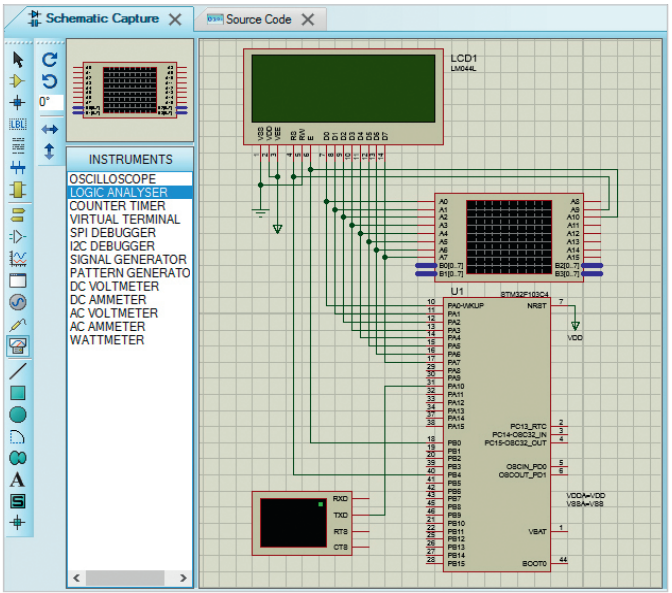

Рис. 14. Выбор логического анализатора на панели INSTRUMENTS и его подключение к схеме ввода/вывода текстовых данных

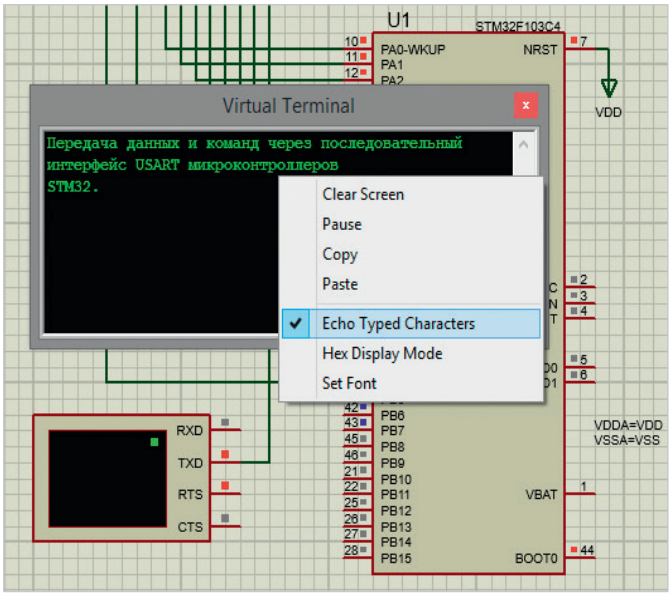

Рис. 15. Настройка параметров отображения символов на экране виртуального терминала

любой операции должны предшествовать команды задержки. Также необходимо обеспечить формирование тактового сигнала на линии E микросхемы LM044L. В нашем примере это сделано программно посредством чередования подачи значений нуля и единицы.

После создания схемы, подключения всех приборов и настройки их параметров переходят к следующему этапу разработки: написанию программного кода управления устройством (в нашем примере на языке C), который в Proteus вводят на вкладке Source Code. В результате его компиляции (при условии отсутствия в коде ошибок) на диске компьютера будет получен исполняемый файл с расширением *.elf, путь к которому автоматически прописывается в окне свойств микроконтроллера в поле Program File.

Завершающий этап работы в редакторе Schematic Capture – запуск процесса моделирования схемы (см. рис. 16), который выполняют кнопкой Run the simulation, расположенной в левом нижнем углу окна редактора или командой основного меню Debug/Run Simulation.

Текст программы инициализации микроконтроллера:

```
#include <stm32f1xx.h> // подключение заголовочного файла
#define F_CPU 2000000 // рабочая частота микроконтроллера
#define baudrate 9600L // скорость обмена данными
void delay (int dly) // подпрограмма формирования задержки
{ int i;
for(; dly>0; dly--)
for ( i=0; i<10000; i++); }

int main() // начало программы
{ RCC->APB2ENR |= RCC_APB2ENR_USART1EN; // включаем тактирование USART1
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
// подсоединение линий порта PB к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPBEN;

GPIOA->CRL = 0x33333333; // линии PA0-PA7 порта PA работают на вывод данных
GPIOB->CRL = 0x33333333; // линии PB0-PB7 порта PB работают на вывод данных
// настройка линии PA10 (RXD) порта PA, биты CNF = 10, биты MODE = 00
GPIOA->CRH = 0x33333833;

// конфигурация USART1
USART1->CR1 = (1<<13); // разрешаем USART1, сбрасываем остальные биты
USART1->BRR = (F_CPU/ (16 * baudrate))*16; // рассчитываем значение для регистра BRR
USART1->CR1 |= (1<<2); // включаем приёмник
USART1->CR2 = 0; // сбрасываем все флаги регистров CR2 и CR3
USART1->CR3 = 0;

// настройка дисплея
// устанавливаем 1 на выводе PB0 микроконтроллера, RS=0 (приём команд)
GPIOB->ODR= (1<<0)|(0<<4);
delay(10); // вызов подпрограммы задержки
// включаем дисплей
GPIOA->ODR =(1<<0)|(1<<1)|(1<<2)|(1<<3)|(0<<4)|(0<<5)|(0<<6)|(0<<7);
// устанавливаем 0 на выводе PB0 микроконтроллера
GPIOB->ODR= (0<<0)|(0<<4);
delay(10);
// установка 8-разрядной шины
GPIOA->ODR = (0<<0)|(0<<1)|(1<<2)|(0<<3)|(1<<4)|(1<<5)|(0<<6)|(0<<7);
// устанавливаем 1 на выводе PB0 микроконтроллера
GPIOB->ODR= (1<<0)|(0<<4);
delay(10);
// очистка дисплея и установка курсора в нулевую позицию
GPIOA->ODR = (1<<0)|(0<<1)|(0<<2)|(0<<3)|(0<<4)|(0<<5)|(0<<6)|(0<<7);
// устанавливаем 0 на выводе PB0 микроконтроллера
GPIOB->ODR= (0<<0)|(0<<4);
delay(10);

while (1) {
GPIOB->ODR = (1<<0)|(1<<4);
delay(10); // RS=1 (приём данных)
while ((USART1->SR & USART_SR_RXNE) == 0) { } // ожидаем данные
char d = USART1->DR; // начинаем прием данных
GPIOA->ODR = d; // отправляем данные на экран дисплея
```

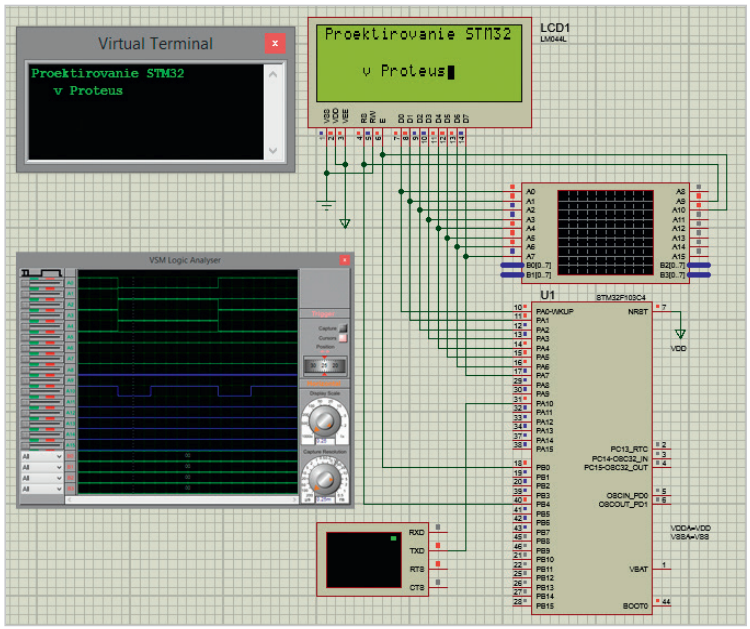

Рис. 16. Результат работы программы управления системой ввода/вывода текстовых данных на базе микроконтроллера STM32F103C4

Watch Window

Name	Address	Value
GPIOA_CRH	0x40010804	0x33333833
MODE0	0x0004	3
CNF0	0x0004	0
MODE1	0x0004	3
CNF1	0x0004	0
MODE2	0x0004	0
CNF2	0x0004	2
MODE3	0x0004	3
CNF3	0x0004	0
MODE4	0x0004	3
CNF4	0x0004	0
MODE5	0x0004	3
CNF5	0x0004	0
MODE6	0x0004	3
CNF6	0x0004	0
MODE7	0x0004	3
CNF7	0x0004	0

Рис. 17. Карта битов регистра GPIOA_CRH

```
  GPIOB->ODR=          (0<<0)|(1<<4);
delay(10); }}
```

Проанализируем работу демонстрационной схемы, представленной на рис. 16. После запуска программа инициализации микроконтроллера включает тактирование модуля USART1 и портов PA и PB, через которые будет вестись приём и передача данных и команд. Далее выполняется запись в соответствующий разряд GPIOx_CRH/CRL нужной комбинации бит для настройки режима работы линий GPIO на приём или передачу информации. Чтение данных из потока ввода виртуального терминала осуществляется определением с помощью второго бита регистра конфигурации GPIOA_CRH режима работы линии PA10. Для линии PA10 в регистре GPIOA_CRH имеется два двухразрядных поля: CNF2 и MODE2. Первое определяет тип работы линии, второе – направление передачи информации по линии. Запись GPIOA->CRH = 0x33333833 в коде программы означает, что линия PA10 порта PA микроконтроллера STM32F103C4 работает на ввод данных – для этой линии в поле MODE записано значение 00 (приём данных), а в поле CNF – значение 10 (вход с «подтягивающим резистором»). Двоичный код 1000 соответствует шестнадцатеричному значению 8, которое записано во второй разряд регистра конфигурации GPIOA_CRH линий 8...15 порта PA (см. рис. 17). Линии PA0...PA9, PA11...PA15 и PB0...PB7 работают на вывод данных, для чего для каждой отдельной линии в поле MODE записаны значения 11 (линия работает на вывод данных с максимальной частотой переключения 50 МГц), а в поле CNF – значение 00 (цифровой выход). Двоичный код 0011 соответствует шестнадцатеричному значению 3, которое записано в соответствующие разряды регистров конфигурации GPIOA_CRH (0x33333833) и GPIOA_CRL (0x33333333), GPIOB_CRL (0x33333333).

Далее программа инициализации микроконтроллера разрешает работу модуля USART1 (команда USART1->CR1 = (1<<13)), включает приёмник (команда USART1->CR1 |= (1<<2)), выполняет расчёт скорости передачи и запись полученного значения в регистр USART1_BRR.

Затем выполняется настройка дисплея, для чего программным путём даны указания микроконтроллеру через порт PA отправить контроллеру микросхемы LM044L кодовые комбинации команд (если на линии PB4 ноль) или данные (если на линии PB4 единица). Для приёма команд/данных в микросхеме LM044L используются линии D0...D7. Управляющий сигнал с линии порта PB4 поступает на вывод RS микросхемы LM044L и подаётся программно. Вывод PB0 микроконтроллера подключён к выводу E микросхемы LM044L и используется для подачи тактовых импульсов.

После запуска симуляции схемы программа инициализации микроконтроллера выводит на линию PB4 логический ноль, который поступает на вывод RS микросхемы LM044L. В результате чего шина D0–D7 переходит в режим приёма следующих команд: включение дисплея (GPIOA->ODR =(1<<0)|(1<<1)|(1<<2)|(1<<3)|(0<<4)|(0<<5)|(0<<6)|(0<<7)), установка 8-разрядной шины (GPIOA->ODR = (0<<0)|(0<<1)|(1<<2)|(0<<3)|(1<<4)|(1<<5)|(0<<6)|(0<<7)), очистка дисплея и установка курсора в нулевую позицию (GPIOA->ODR = (1<<0)|(0<<1)|(0<<2)|(0<<3)|(0<<4)|(0<<5)|(0<<6)|(0<<7)). Далее программа инициализации микроконтроллера выводит на линию PB4 логическую единицу, что переводит шину D0...D7 микросхемы LM044L в режим приёма данных (команда GPIOA->ODR = d), запись которых выполняется побайтно в цикле в регистр USART1_DR (команда d = USART1->DR) после установки в лог. «1» флага RXNE регистра USART1_SR. При этом на вывод E непрерывно подаётся тактовый сигнал, по заднему фронту которого микросхема LM044L считывает информацию (команды/данные). Таким образом, на экран микросхемы LM044L посимвольно выводится строка, двоичные коды символов которой были поданы на шину D0...D7 (см. рис. 16). Символ отображается на экране дисплея после его ввода на экране виртуального терминала. Временны́е диаграммы работы схемы ввода/вывода текстовых данных на базе микроконтроллера STM32F103C4 представлены на рис. 16.

Работа с двумя USART

Работа с двумя модулями USART показана на рис. 18. Ниже представлен текст программы инициализации микроконтроллера STM32F103C4, которая управ-

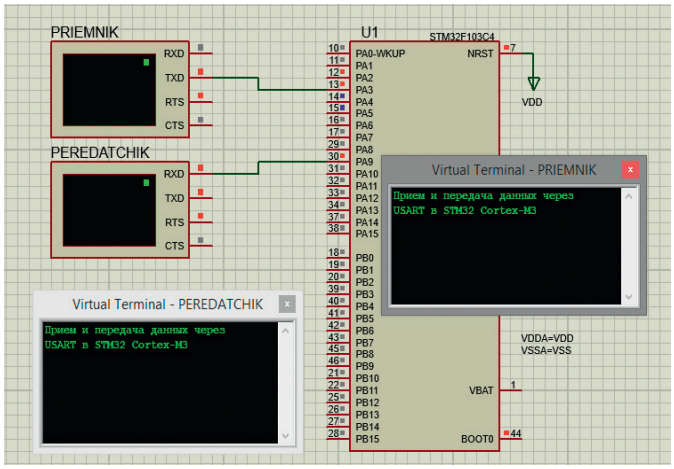

Рис. 18. Приём данных через USART2 и их передача через USART1 в микроконтроллере STM32F103C4

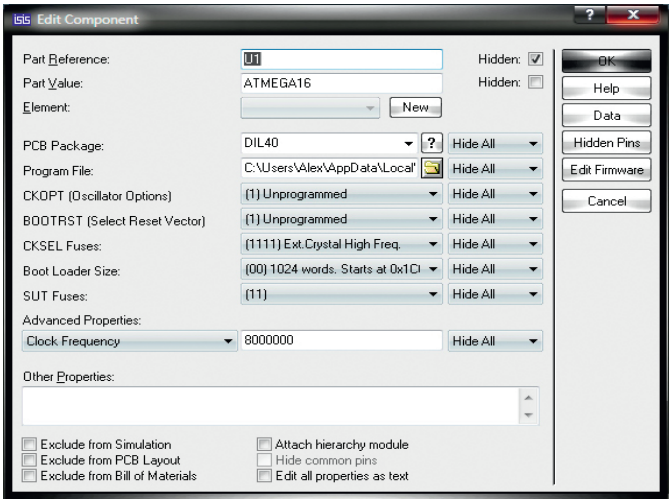

Рис. 19. Окно настроек микроконтроллера ATmega16

ляет приёмом данных по USART2 через виртуальный терминал PRIEMNIK и их выводом через USART1 на экран терминала PEREDATCHIK. Текст программы инициализации микроконтроллера:

```
#include <stm32f1xx.h> // подключение заголовочного файла
#define F_CPU 2000000 // рабочая частота контроллера
#define baudrate 9600L // скорость обмена данными

int main() // начало программы
{ RCC->APB2ENR |= RCC_APB2ENR_USART1EN; // включаем тактирование USART1
RCC->APB1ENR |= RCC_APB1ENR_USART2EN; // включаем тактирование USART2
// подсоединение линий порта PA к шине APB2
RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;

// настройка линии PA3 (RXD) порта PA, биты CNF = 10, биты MODE = 00
GPIOA->CRL = 0x00008000;
// настройка линии PA9 (TXD) порта PA, биты CNF = 10, биты MODE = 01
GPIOA->CRH = 0x00000090;

// конфигурация USART2
USART2->CR1 = (1<<13); // разрешаем USART2, сбрасываем остальные биты
USART2->BRR = (F_CPU/ (16 * baudrate))*16; // рассчитываем значение для регистра BRR
USART2->CR1 |= (1<<2); // включаем приёмник
USART2->CR2 = 0; // сбрасываем все флаги регистров CR2 и CR3
USART2->CR3 = 0;

// конфигурация USART1
USART1->CR1 = (1<<13); // разрешаем USART1, сбрасываем остальные биты
USART1->BRR = (F_CPU/ (16 * baudrate))*16; // рассчитываем значение для регистра BRR
USART1->CR1 |= (1<<3); // включаем передатчик
USART1->CR2 = 0; // сбрасываем все флаги регистров CR2 и CR3
USART1->CR3 = 0;

while (1) { // бесконечный цикл
while ((USART2->SR & USART_SR_RXNE) == 0) { } // ожидаем данные
char d = USART2->DR; // начинаем приём данных с экрана первого терминала
while ( USART1->SR == ((0<<6)|(0<<7)) ) { } // ожидаем, когда очистится буфер передачи
// помещаем данные в буфер, начинаем передачу на экран второго терминала
USART1->DR = d;
}}
```

Работа с универсальным синхронно/асинхронным приёмопередатчиком USART в микроконтроллерах Mega в Proteus

Рассмотрим процесс моделирования схем с использованием микроконтроллеров AVR семейства Mega на примере микросхем ATmega128 и ATmega16. Все микроконтроллеры семейства Mega имеют в своём составе от одного до четырёх модулей универсального синхронно/асинхронного приёмопередатчика USART. В микроконтроллере ATmega16 присутствует один такой модуль, а в ATmega128 – два.

Выводы микроконтроллера, используемые модулями USART, являются линиями ввода/вывода общего назначения. К примеру, в микроконтроллере ATmega16 модулем USART используются линии PD0 (RXD) – вход USART, PD1 (TXD) – выход USART, PB0 (XCK) – вход/выход внешнего тактового сигнала USART.

Модуль состоит из трёх основных частей: блока тактирования, блока передатчика и блока приёмника. Буферные регистры приёмника и передатчика располагаются по одному адресу пространства ввода/вывода и обозначаются как регистр данных UDR. В этом регистре хранятся младшие 8 бит принимаемых и передаваемых данных. При чтении регистра UDR выполняется обращение к буферному регистру приёмника, при записи – к буферному регистру передатчика.

Для управления модулем USART используются три регистра: UCSRA, UCSRB, UCSRC. Работа передатчика разрешается установкой в лог. «1» бита TXEN регистра UCSRB. При установке бита вывод TXD подключается к передатчику USART и начинает функционировать как выход, независимо от установок регистров управления портом. Если используется синхронный режим работы, то переопределяется также функционирование вывода XCK. Передача инициируется записью передаваемых данных в буферный регистр передатчика – регистр данных UDR. После этого данные пересылаются из регистра UDR в сдвиговый регистр передатчика. После пересылки слова данных в сдвиговый регистр флаг UDRE регистра UCSRA устанавливается в 1, что означает готовность передатчика к получению нового слова данных.

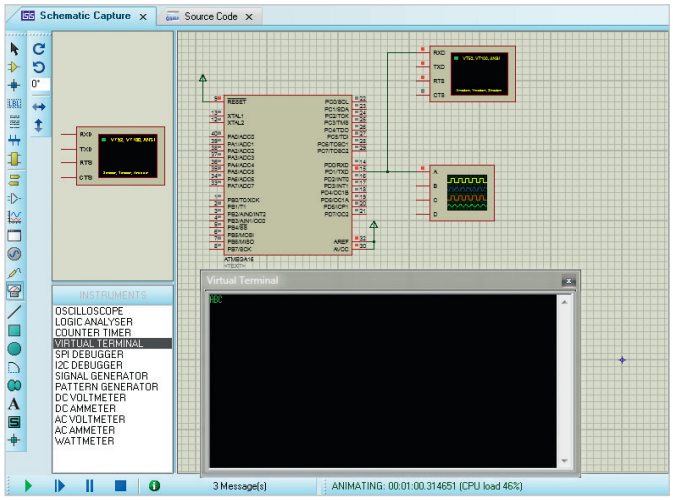

Рис. 20. Схема передачи данных при помощи модуля USART микроконтроллера ATmega16

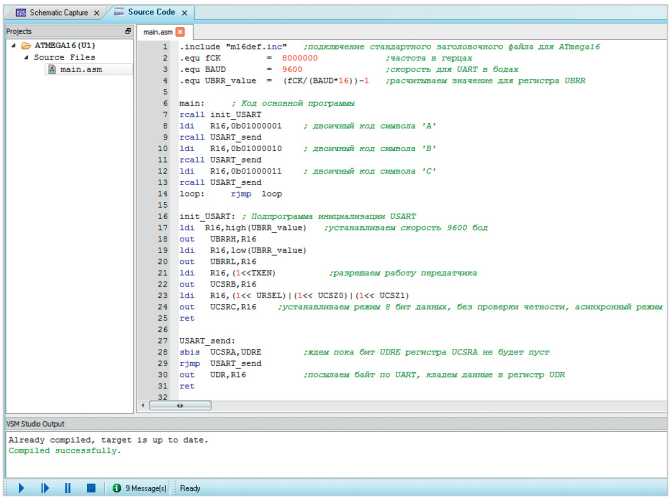

Рис. 21. Код программы инициализации микроконтроллера ATmega16 на вкладке Source Code

В этом состоянии флаг остаётся до следующей записи в буфер.

Выключение передатчика осуществляется сбросом бита TXEN регистра UCSRB. Если в момент выполнения этой команды осуществлялась передача, сброс бита произойдёт только после завершения текущей и отложенной передач, то есть после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD может использоваться как контакт ввода/вывода общего назначения.

Работа приёмника разрешается установкой бита RXEN регистра UCSRB. При установке бита вывод RXD подключается к приёмнику USART и начинает функционировать как вход, независимо от установок регистров управления портом. Если используется синхронный режим работы, переопределяется также функционирование вывода XCK.

Выключение приёмника осуществляется сбросом бита RXEN регистра UCSRB. В отличие от передатчика, приёмник выключается сразу же после сброса бита, а значит, кадр, принимаемый в этот момент, теряется. Кроме того, при выключении приёмника очищается его буфер, то есть теряются также все непрочитанные данные. При выключенном приёмнике вывод RXD может использоваться как контакт ввода/вывода общего назначения.

Рассмотрим работу модуля USART на конкретном примере. Передадим программным способом на экран виртуального терминала комбинацию символов «ABC». Для этого создадим в Proteus новый проект с использованием микроконтроллера ATmega16 и добавим в рабочее поле виртуальный терминал, а также виртуальный осциллограф для просмотра осциллограммы работы USART.

Подсоединим вывод TXD микроконтроллера к выводу RXD виртуального терминала, а также к каналу A осциллографа. В окне настроек микроконтроллера Edit Component установим следующие параметры (см. рис. 19):

- поле CKOPT (Oscillator Options) – (1) Unprogrammed;
- поле BOOTRST (Select Reset Vector) – (1) Unprogrammed;
- поле CKSEL Fuses – (1111) Ext.Crystal High Freq.;
- поле Boot Loader Size – (00) 1024 words. Starts at 0x1C00;
- поле SUT Fuses – (11);
- поле Advanced Properties – Clock Frequency 8000000.

Окно настроек открывают двойным щелчком левой кнопки мыши по выбранному на схеме микроконтроллеру.

Для проверки работы собранной схемы (см. рис. 20) на языке программирования ассемблер была написана следующая программа (см. рис. 21):

```
.include «m16def.inc» ; подключение стандартного заголовочного файла для ATmega16
.equ fCK = 8000000 ; частота в герцах
.equ BAUD = 9600 ; скорость для USART в бодах
.equ UBRR_value = (fCK/(BAUD×16)) – 1 ; рассчитываем значение для регистра UBRR
main: ; Код основной программы
rcall init_USART
ldi R16,0b01000001 ; двоичный код символа ‘A’
rcall USART_send
ldi R16,0b01000010 ; двоичный код символа ‘B’
rcall USART_send
ldi R16,0b01000011 ; двоичный код символа ‘C’
rcall USART_send
loop: rjmp loop
init_USART: ; Подпрограмма инициализации USART
ldi R16,high(UBRR_value) ; устанавливаем скорость 9600 бод
out UBRRH,R16
ldi R16,low(UBRR_value)
out UBRRL,R16
ldi R16,(1<<TXEN) ; разрешаем работу передатчика
out UCSRB,R16
ldi R16,(1<< URSEL)|(1<< UCSZ0)|(1<< UCSZ1)
out UCSRC,R16 ; устанавливаем режим 8 бит данных, без проверки чётности,
;асинхронный режим
ret
USART_send:
sbis UCSRA,UDRE ; ждём пока бит UDRE регистра UCSRA не будет пуст
rjmp USART_send
out UDR,R16 ; посылаем байт по USART, кладём данные в регистр UDR
ret
```

После того как в рабочей области проекта собрана схема, а на вкладке Source Code введён код программы, можно запускать моделирование. Как видно из рис. 20, разработанный проект функционирует верно: на экран виртуального терминала была выведена указанная в коде программы комбинация символов. Осциллограмма работы USART показана на рис. 22.

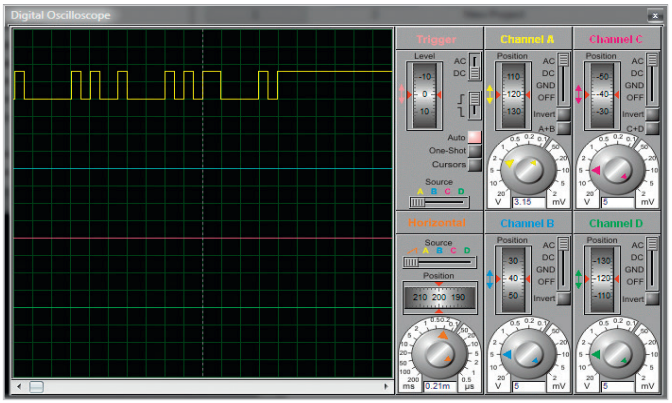

Рис. 22. Осциллограмма работы модуля USART микроконтроллера ATmega16

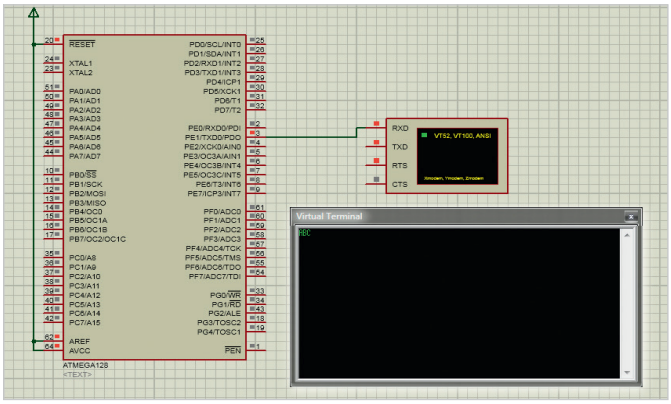

Рис. 23. Схема передачи данных при помощи модуля USART0 микроконтроллера ATmega128

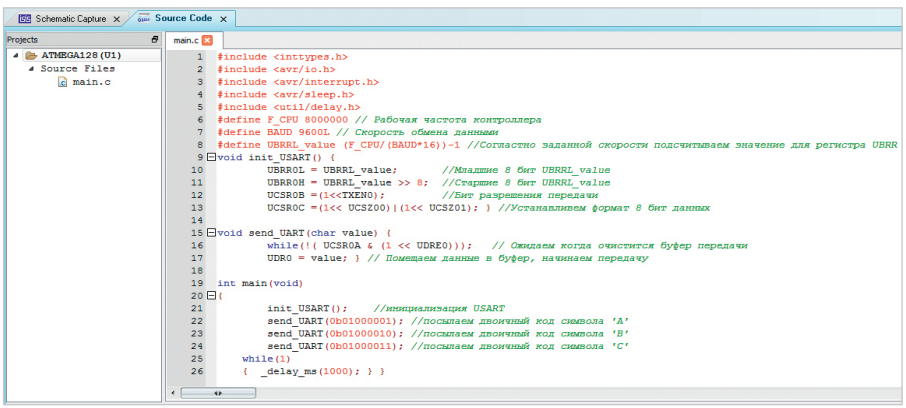

Рис. 24. Код программы инициализации микроконтроллера ATmega128 на вкладке Source Code

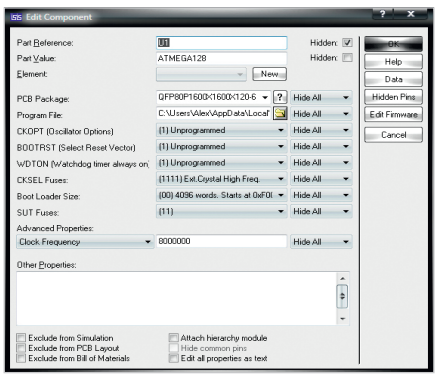

Рис. 25. Окно настроек микроконтроллера ATmega128

Рассмотрим передачу данных по интерфейсу USART в микроконтроллере ATmega128, для чего напишем аналогичную программу передачи символов «ABC» на экран виртуального терминала на языке программирования C.

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>
#define F_CPU 8000000 // Рабочая частота контроллера
#define BAUD 9600L // Скорость обмена данными
#define UBRRL_value (F_CPU/(BAUD×16)) - 1 // Согласно заданной скорости
//подсчитываем значение для регистра UBRR
void init_USART() {
UBRR0L = UBRRL_value; // Младшие 8 бит UBRRL_value
UBRR0H = UBRRL_value >> 8; // Старшие 8 бит UBRRL_value
UCSR0B = (1<<TXEN0); // Бит разрешения передачи
UCSR0C = (1<< UCSZ00)|(1<< UCSZ01); } // Устанавливаем формат 8 бит данных
void send_USART(char value) {
while(!( UCSR0A & (1 << UDRE0))); // Ожидаем когда очистится буфер передачи
UDR0 = value; } // Помещаем данные в буфер, начинаем передачу
int main(void)
{
init_USART(); // инициализация USART
send_USART(0b01000001); // посылаем двоичный код символа 'A'
send_USART(0b01000010); // посылаем двоичный код символа 'B'
send_USART(0b01000011); // посылаем двоичный код символа 'C'
while(1)
{ _delay_ms(1000); } }
```

Здесь необходимо отметить, что в микроконтроллере ATmega128 два модуля USART: USART0, USART1. Таким образом, при написании программного кода необходимо указывать, к регистрам какого модуля USART мы обращаемся. Если при работе с интерфейсом USART микроконтроллера ATmega16 мы обращались к регистру данных по имени UDR, то при работе, к примеру, с модулем USART0 микроконтроллера ATmega128 к регистру данных необходимо обращаться по имени UDR0.

На рис. 23 показана схема передачи данных при помощи модуля USART0 микроконтроллера ATmega128. Код программы инициализации микроконтроллера на вкладке Source Code представлен на рис. 24. На рис. 25 показано окно настроек микроконтроллера ATmega128.

Литература

1. Proteus VSM Help, Labcenter Electronics, 2020.
2. STM32F103x4, STM32F103x6 MCU Datasheet. STMicroelectronics. 2009.
3. HD44780U (LCD-II) (Dot Matrix Liquid Crystal Display Controller/Driver). Hitachi, Ltd. 1998.
4. STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced ARM-based 32-bit MCUs. Reference manual. STMicroelectronics. 2010.

НОВОСТИ МИРА

В России готовится строительство линейки суперкомпьютеров на «Эльбрусах» до 100 петафлопс

К 2026 г. в России может быть создан суперкомпьютер 100-петафлопсной производительности, построенный на разрабатываемых сейчас 32-ядерных отечественных процессорах «Эльбрус-32С». На сегодняшний день подобная производительность могла бы позволить попасть системе на пятую строчку в списке мощнейших вычислителей планеты.

В распоряжении редакции оказался слайд, по всем признакам являющийся фрагментом презентации суперкомпьютерной компании РСК, на котором отображена «дорожная карта» развития технологий этой организации на «Эльбрусах» до 2027 г. Наиболее отдалённой от сегодняшнего дня разработкой на этой карте указана суперЭВМ упомянутой производительности на чипах «Эльбрус-32С». Её появление запланировано на 2026...2027 г.

Суперкомпьютер займет 165 серверных шкафов, в которых разместится 16,5 тыс. лезвий суммарно с 33 тыс. процессорами. Судя по слайду, в нем предполагается использовать интерконнект разработки Российского федерального ядерного центра – Всероссийского НИИ экспериментальной физики в Сарове (РФЯЦ-ВНИИЭФ; входит в «Росатом»). О том, что у этого института есть решения для интерконнекта, в декабре 2018 г. указывал директор Института программных систем им. А. К. Айламазяна РАН Сергей Абрамов. В готовности этого интерконнекта на сегодняшний день CNews заверили представители МЦСТ – компании, разрабатывающей «Эльбрусы».

Что касается упомянутого процессора «Эльбрус-32С», то он пока не готов, но должен появиться к 2025 г. О его создании стало известно в конце октября 2020 г. Этот 32-ядерный чип будет реализован по топологии 6 или 7 нм.

В РСК подлинность слайда CNews не опровергли, однако к моменту публикации как-либо прокомментировать проект не смогли.

Представитель МЦСТ Максим Горшенин в разговоре с CNews отметил, что представленная дорожная карта вполне соответствует срокам появления серийных микропроцессоров его компании. «МЦСТ видит спрос на высокопроизводительные вычислительные кластеры, – говорит он. – Совместно с компанией РСК давно идёт работа по созданию высоконагруженных систем с высокой вычислительной плотностью. Есть заказчики, которые тестируют наши решения».

При этом Горшенин подчеркивает, что не нужно воспринимать указанные в плане даты как абсолютные константы. «Это видение РСК того, как и когда можно реализовать разработки, – добавляет он. – МЦСТ с этим видением согласна. Но речь не идёт о том, что это все жёстко зафиксировано и точно появится в прописанные сроки».

Отметим, что на сегодняшний день 100-петафлопсная производительность приблизительно соответствует пятой строчке в списке мощнейших вычислителей планеты топ-500. Правда нужно понимать, что этот рейтинг формируется не по пиковой произ-

НОВОСТИ МИРА

водительности, а по результатам стандартного теста Linpack. Их значения несколько отличаются, причём не во вполне заранее предсказуемой пропорции.

cnews.ru

Дефицит чипов начнёт ослабевать в 2022 году, считают в AMD

Дефицит чипов начнёт ослабевать в 2022 году, заявила Лиза Су (Lisa Su), генеральный директор компании Advanced Micro Devices (AMD), известной по производству микропроцессоров для персональных компьютеров, серверов и другой электроники.

Свой прогноз Су озвучила на конференции Code Conference в Беверли-Хиллз (США). По словам главы AMD, участники полупроводниковой отрасли активно инвестируют в расширение производственных мощностей. Так, в 2021 году ожидается запуск 20 новых заводов по изготовлению чипов, в 2022-м может быть построено столько же предприятий.

«Пандемия подняла спрос [на полупроводники – прим. DailyComm] на новый уровень… Никто не ожидал столь большого спроса», – сообщила Лиза Су.

Она считает, что в следующем полугодии дефицит компонентов ещё сохранится, а вот во второй половине 2022 года его остроту удастся снизить. Попутно глава AMD высказалась в поддержку инициатив американских законодателей по субсидированию развития национальной полупроводниковой отрасли.

В сентябре 2021 года глава Tesla и SpaceX Илон Маск (Elon Musk) предсказал, когда закончится дефицит микросхем, вызвавший кризис в автомобильной промышленности. Закрыть всемирную проблему нехватки чипов поможет строительство новых заводов по производству полупроводников. По словам миллиардера, есть хороший потенциал для того, чтобы обеспечить поставки чипов уже к 2022 году.

Ранее деловое издание Nikkei сообщило, что полупроводниковая отрасль попала в замкнутый круг. Чтобы решить проблему дефицита чипов, нужно увеличить производственные мощности по выпуску микросхем. Но производители оборудования для изготовления полупроводников сами страдают от нехватки мощностей.

Проблема затронула минимум четыре важнейших категории полупроводникового оборудования. Среди них – установки для монтажа кристаллов, установки для резки полупроводниковых пластин, установки для проверки полупроводниковых кристаллов и системы лазерного сверления.

Консалтинговая компания AlixPartners в сентябре 2021 года вдвое ухудшила майский прогноз и назвала текущий дефицит поставок самым длинным в истории отрасли. По данным аналитиков, всего в 2021 году мировые автопроизводители выпустят на 7,7 млн автомобилей меньше, чем запланировано.

Из-за постоянного ухудшения ситуации, прогноз неоднократно корректировался в отрицательную сторону. В январе в отрасли ожидали годовое недополучения 61 млрд долларов, а в мае – 110 млрд долларов. За 4 месяца прогноз снова ухудшился более чем на 90% – теперь компании могут потерять до 210 млрд долларов по итогам 2021 года.

russianelectronics.ru