DOI 10.37882/2223-2966.2024.04.40

СПОСОБЫ ПРОГРАММНОЙ РЕАЛИЗАЦИИ РАБОТЫ UART

METHODS OF SOFTWARE IMPLEMENTATION OF UART OPERATION

S. Shchegolev
T. Efremova
A. Motkov

Summary. The article describes the operation of the serial asynchronous UART interface. The UART is used to transfer data between the microcontroller and the computer. Setting up the UART includes selecting the synchronization source and determining the operating mode. To set the UART operation mode, you need to change the values of the SM0 and SM1 bits in the SCON register. Synchronization can be carried out from Timer 3 or Timer 1. To determine the information transfer rate via UART, it is necessary to calculate the value of the counter register TH1. The timer is started automatically, and data is transmitted through the SBUF register. The TI flag signals the completion of the transfer, and you can arrange for TI to be checked for zero in the loop. To optimize the execution time of the program, you can change the section of the program that sends a byte of data.

Keywords: microcontroller, data transfer protocol, timer, command, program, UART, transfer rate, operating mode.

Щеголев Сергей Сергеевич
Доцент, кандидат технических наук, Балаковский инженерно-технологический институт — филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ», Балаково, Россия
SSShchegolev@mephi.ru
Ефремова Татьяна Александровна
Доцент, кандидат технических наук, Балаковский инженерно-технологический институт — филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ», Балаково, Россия
TAEfremova@mephi.ru
Мотков Александр Геннадьевич
Старший преподаватель, Балаковский инженерно-технологический институт — филиал федерального государственного автономного образовательного учреждения высшего образования «Национальный исследовательский ядерный университет «МИФИ», Балаково, Россия
AGMotkov@mephi.ru

Аннотация. Статья описывает работу последовательного асинхронного интерфейса UART. UART используется для передачи данных между микроконтроллером и компьютером. Настройка UART включает выбор источника синхронизации и определение режима работы. Для установки режима работы UART необходимо изменить значения битов SM0 и SM1 в регистре SCON. Синхронизация может осуществляться от Таймера 3 или Таймера 1. Для определения скорости передачи информации по UART необходимо рассчитать значение регистра счетчика TH1. Запуск таймера происходит автоматически, а передача данных осуществляется через регистр SBUF. Флаг TI сигнализирует о завершении передачи, и можно организовать проверку TI на равенство нулю в цикле. Для оптимизации времени выполнения программы можно изменить участок программы, отправляющий байт данных.

Ключевые слова: микроконтроллер, протокол передачи данных, таймер, команда, программа, UART, скорость передачи, режим работы.

Узел вычислительных устройств, называемый универсальным асинхронным приёмопередатчиком, предназначен для связи с другими цифровыми устройствами. На сегодняшний день самым распространенным физическим протоколом передачи данных является протокол UART. По умолчанию, последовательный порт микроконтроллера ADuC842 синхронизируется от Таймера 1. Скорость передачи данных определяется временем срабатывания таймера по формуле [1]:

$$BR = \frac{1}{32 \cdot T_T}, \quad (1)$$

где T_T — время срабатывания таймера.

Для того, чтобы установить режим работы таймера в режиме с автоперезагрузкой (режим 2), необходимо записать бинарную комбинацию 0010b в старшие 4 бита регистра TMOD. В этом случае скорость передачи данных будет определяться другой формулой:

$$BR = \frac{f_{core}}{32 \cdot \left(256 - TH1\right)}, \quad (2)$$

где f_{core} — частота ядра микропроцессора; $TH1$ — содержимое регистра данных $TH1$.

Значение регистра $TH1$, которое обеспечивает требуемую скорость, может быть найдено из формулы:

$$TH1 = 256 - \frac{f_{core}}{32 \cdot BR} \quad (3)$$

Формула (3) позволяет получить результат вычисления, округленный до ближайшего целого, для *TH*1, который обеспечивает требуемую скорость.

Использование Таймера 1 для синхронизации UART не всегда позволяет достичь требуемой частоты с достаточной точностью. Если, например, при тактовой частоте ядра микропроцессора 2097 кГц (значение для ADuC842 по умолчанию) необходимо получить скорость передачи 19.2 кбит/с, то значение *TH*1 можно найти по формуле 3:

$$TH1 = 256 - \frac{2097}{32 \cdot 19,2} = 252,59 \approx 252. \quad (4)$$

Используя полученное значение *TH*1, рассчитаем реальную скорость передачи UART:

$$BR = \frac{2097}{32 \cdot (256 - 252)} = 16,38 \text{ (кбит/с)} \quad (5)$$

Проблема невозможности передачи данных возникла из-за того, что реальная скорость передачи UART оказалась на 14 % меньше требуемой. Для решения этой проблемы был добавлен специальный таймер 3, который предназначен для высокоточной синхронизации UART в широком диапазоне частот. Таймер 3 представляет собой набор настраиваемых делителей тактовой частоты ядра. Для управления таймером 3 используются два регистра специальных функций — *T3CON* и *T3FD*. Используя полученное значение *TH*1, можно рассчитать реальную скорость передачи UART, которая равна 16,38 (кбит/с), что находится ниже требуемого значения.

Для синхронизации UART используется регистр *T3CON*, содержащий бит *T3EN*. Если он установлен в единицу, то синхронизация происходит от Таймера 3, иначе — от Таймера 1. Двоичный делитель DIV определяется младшими тремя битами этого же регистра. Регистр T3FD настраивает дробный коэффициент деления.

T3CON — это регистр конфигурации Таймера 3, который находится по адресу 0x9E в SFR. Значение регистра после подачи питания равно 0x00. Он не имеет побитовой адресации.

Регистр *T3FD* — это регистр Таймера 3, который находится по адресу 0x9D в SFR. Значение регистра после подачи питания равно 0x00. Он также не имеет побитовой адресации.

Для расчета результирующей скорости последовательного порта с использованием структурной схемы Таймера 3 необходимо записать аналитическое выражение. Оно может быть представлено следующим образом:

$$BR = \frac{2 \cdot f_{core}}{2^{DIV-1} \cdot (T3FD + 64)}, \quad (6)$$

где f_{core} — частота ядра микроконтроллера.

Значение делителя *DIV* можно определить с помощью формулы 7, а дробный делитель *T3FD* — по формуле 8. При этом значение *DIV* следует округлить до целого вниз, а значение *T3FD* — до ближайшего целого [2]. Формула для определения *DIV* выглядит следующим образом:

$$DIV = log_2 \frac{f_{core}}{16 \cdot BR}. \quad (7)$$

Формула для определения *T3FD*:

$$T3FD = \frac{2 \cdot f_{core}}{2^{DIV-1} \cdot BR} - 64. \quad (8)$$

Для предыдущего примера необходимо рассчитать параметры конфигурации Таймера 3, при тактовой частоте ядра микропроцессора 2097 кГц и требуемой скорости передачи 19.2 кбит/с. Для этого применяем формулы:

$$DIV = log_2 \frac{2097}{16 \cdot 19,2} = 2,771 \approx 2 \quad (9)$$

$$T3FD = \frac{2 \cdot 2097}{2^{2-1} \cdot 19,2} - 64 = 45,219 \approx 45 \quad (10)$$

$$BR = \frac{2 \cdot 2097}{2^{2-1} \cdot (45 + 64)} = 19.239 \text{ (кбит/с)} \quad (11)$$

Таким образом, ошибка установления скорости составляет всего 0.2 %. Перед первым обращением к приемо-передатчику UART необходимо настроить последовательный порт: выбрать и настроить источник синхронизации, определить режим работы.

Для установки режима работы UART необходимо изменить значения битов SM0 и SM1, находящихся в регистре SCON. Этот регистр имеет как байтовую, так и битовую адресацию, поэтому его можно настроить несколькими способами: либо записать требуемое число в регистр SCON, либо установить каждый бит отдельно. Если бит T3EN регистра *T3CON* установлен в логическую единицу, то источником синхронизации будет Таймер 3, если же этот бит равен нулю (T3EN = 0 по умолчанию), то синхронизация будет происходить от Таймера 1. Если необходимо использовать Таймер 1, его нужно настроить на работу в режиме 2 (свободнобегущий таймер с автоперезагрузкой), для чего следует записать двоичную комбинацию 0010b в старшие четыре бита регистра SMOD.

Для определения скорости передачи информации по UART необходимо рассчитать значение регистра

счетчика *TH*1 по формуле 3. Для запуска таймера необходимо записать логическую единицу в бит TR1 регистра *TMOD* после записи *TH*1. Если синхронизация осуществляется от Таймера 3, то делители DIV и *T3FD* рассчитываются по формулам 5 и 6 соответственно. При записи делителя *T3FD* в регистр *T3FD*, необходимо определить делитель *DIV* младшими тремя битами регистра *T3CON*, при этом в старший бит этого регистра (*T3EN*) должна быть записана логическая единица. Запуск таймера происходит автоматически [3].

Для начала отправки данных по UART необходимо выполнить любую команду, результат которой будет записан в регистр *SBUF*. Например, чтобы отправить символ «E», можно использовать следующий код: *SBUF* = 0x45. Однако каждый раз не обязательно использовать таблицу *ASCII* для определения кода символа. В языке программирования «Си» есть удобный инструмент — достаточно взять нужный символ в апострофы, и компилятор интерпретирует это как код символа. Таким образом, можно отправить символ «E» следующим образом: *SBUF* = 'E'. Если требуется отправить несколько символов, необходимо дождаться отправки предыдущего символа, прежде чем записывать следующий код в регистр *SBUF*.

Флаг TI регистра *TCON* сигнализирует о завершении передачи. Как только передача завершена, в бит TI записывается логическая единица. Можно организовать проверку TI на равенство нулю в цикле и отправлять следующий байт только тогда, когда TI станет равен единице.

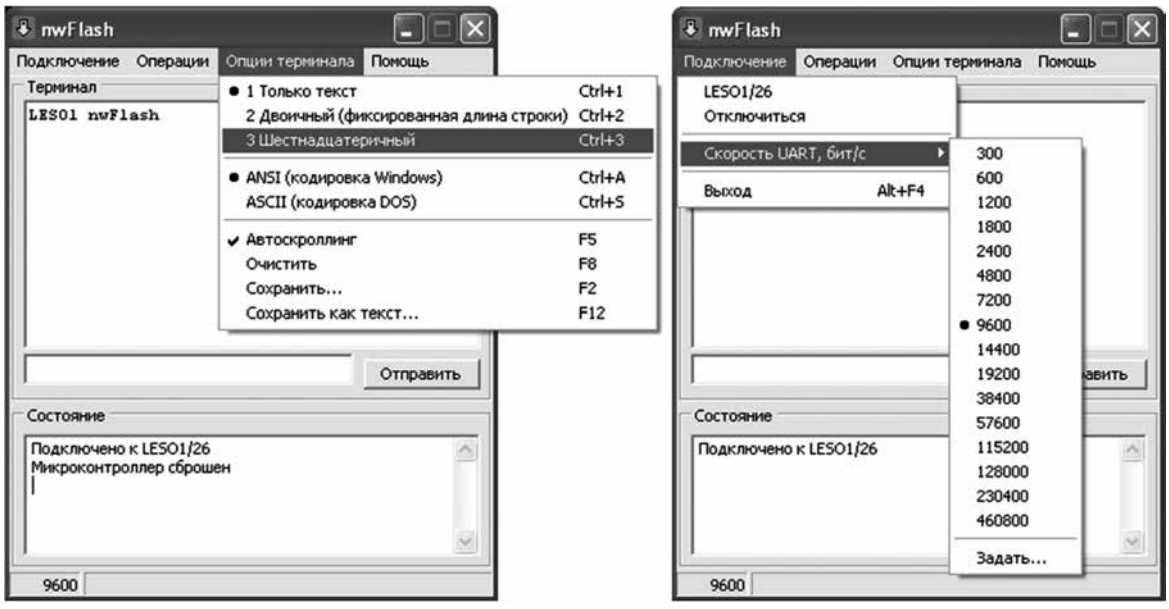
Для приложений, где необходимо оптимизировать время выполнения, можно изменить участок программы, отправляющий байт данных. Вместо ожидания полной отправки байта и освобождения буфера, можно без задержки приступать к выполнению следующей программы. Однако перед отправкой следующего байта необходимо проверить, что буфер освободился и передатчик готов к работе. Для этого можно использовать следующие команды: сначала ждем завершения приема байта с помощью команды while(!RI), затем считываем принятый байт в переменную cmd с помощью команды cmd = SBUF, и сбрасываем флаг приема с помощью команды RI = 0 [4]:

```
while(!RI); // ждем завершение приема байта
cmd = SBUF; // считываем принятый байт в переменную cmd
RI = 0; // сброс флага приема
```

Для заполнения буфера и начала передачи необходимо подождать, пока буфер передачи не освободится (если занят), что достигается с помощью команды while(!TI). Затем заполнение буфера и начало передачи выполняются с помощью *SBUF* = 0x45. Для сброса флага передачи в нуль используется команда TI = 0. Такой вариант реализации позволяет устранить паузы на выполнение программы между передачами отдельных байтов:

```
SBUF = 0x45; // отправить символ "E"
while(!TI); // пока TI равен нулю, выполнять пустой цикл
TI = 0; // сбросить флаг для следующей передачи
```

При написании программы для микроконтроллера необходимо учитывать, что она должна выполняться до отключения питания устройства и не может быть завершена. Для этого программа должна содержать бесконечный цикл [5].



а) б)

Рис. 1. Настройка опций терминала (а) и настройка скорости UART (б)

Для подключения учебного лабораторного стенда к персональному компьютеру используется микросхема преобразователя интерфейсов USB-UART.

В программе загрузчика nwFlash реализован терминал для связи с микроконтроллером. Терминал позволяет отправлять и принимать информацию через последовательный порт, а также отображать принятую информацию. Настройка терминала производится в пункте «Опции терминала» (рисунок 1,а) в главном меню. Для работы с учебным стендом необходимо установить требуемую скорость подключения в меню «подключение», как показано на рисунке1,б.

В графическом виде алгоритм программы работы последовательного асинхронного интерфейса UART представлен на рисунке 2.

Основная программа включает 3 подпрограммы и реализует прием и передачу символов через порт UART. Для синхронизации используется Таймер3, а скорость передачи данных составляет 9600 бит/с [6].

При запуске программы появляется сообщение «Введите символ.», которое отправляет подпрограмма приветствия на компьютер. Затем запускается подпрограмма приема символа, которая считывает символ, введенный пользователем с клавиатуры, и записывает его в переменную cmd. В конце запускается подпрограмма передачи полученного символа, которая выводит сообщение «Вы ввели символ: » в окно терминала, а затем отображает введенный ранее пользователем символ.

Текст программы на языке С-51 [7]:

```
#include <stdio.h>
// Ob'yavimperemenniepobytno
sbit   TI =0x99;          // Flag priema
sbit   RI =0x98;          // Flag peredachi
sfr    T3CON=0x9E;     //
Opredelyaettaimersinhronizatsii
sfr    SBUF =0x99;       // Bufer
sfr    T3FD =0x9D;       // Delitel' dlya Taimera3
sfr    PLLCON =0xD7;   // Delitel'
dlyaTaktovogoGeneratora
sfr    SCON =0x98;       // Opredelyaetregimraboty UART
charcmd;
voidPrivetstvie(){
while(!TI); TI=0;SBUF=0x0A;while(!TI);TI=0;SBUF='B';
while(!TI); TI=0;SBUF='в';while(!TI); TI=0;SBUF='е';
while(!TI); TI=0;SBUF='д';while(!TI); TI=0;SBUF='и';
while(!TI); TI=0;SBUF='т';while(!TI); TI=0;SBUF='е';
while(!TI); TI=0;SBUF='';while(!TI); TI=0;SBUF='с';
while(!TI); TI=0;SBUF='и';while(!TI); TI=0;SBUF='м';
while(!TI); TI=0;SBUF='в';while(!TI); TI=0;SBUF='о';
while(!TI); TI=0;SBUF='л';while(!TI); TI=0;SBUF=':';
while(!TI); TI=0;SBUF='';while(!TI); TI=0;SBUF='';
```

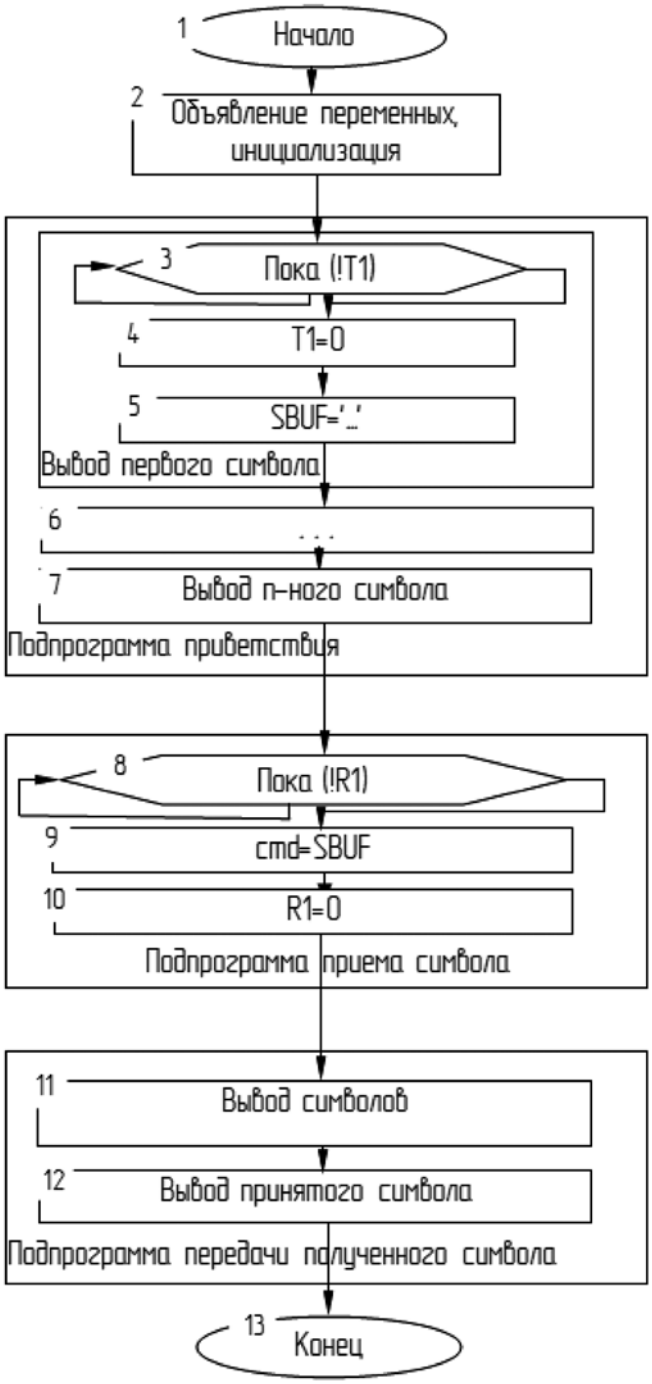


Рис. 2. Алгоритм программы работы порта UART

```
}
Priem(){
while(!RI);
cmd=SBUF;
      RI=0;
}
voidPeredacha(){
while(!TI); TI=0; SBUF=0x0A;while(!TI); TI=0;SBUF='B';
while(!TI); TI=0;SBUF='ы';while(!TI); TI=0;SBUF='';
while(!TI); TI=0;SBUF='в';while(!TI); TI=0;SBUF='в';
```

```
while(!TI); TI=0;SBUF='е';while(!TI); TI=0;SBUF='л';
while(!TI); TI=0;SBUF='и';while(!TI); TI=0;SBUF='';
while(!TI); TI=0;SBUF='с';while(!TI); TI=0;SBUF='и';
while(!TI); TI=0;SBUF='м';while(!TI); TI=0;SBUF='в';
while(!TI); TI=0;SBUF='о';while(!TI); TI=0;SBUF='л';
while(!TI); TI=0;SBUF=':';while(!TI); TI=0;SBUF='';
while(!TI); TI=0;SBUF='';
SBUF=cmd;
while(!TI);
        TI=0;
}
Main(){
        T3CON =0x86-(PLLCON &0x03);
        T3FD =0x02D;
        SCON=0x052;
Privetstvie();
Priem();
Peredacha();
}
```

Результат работы программы представлен на рисунке 3.

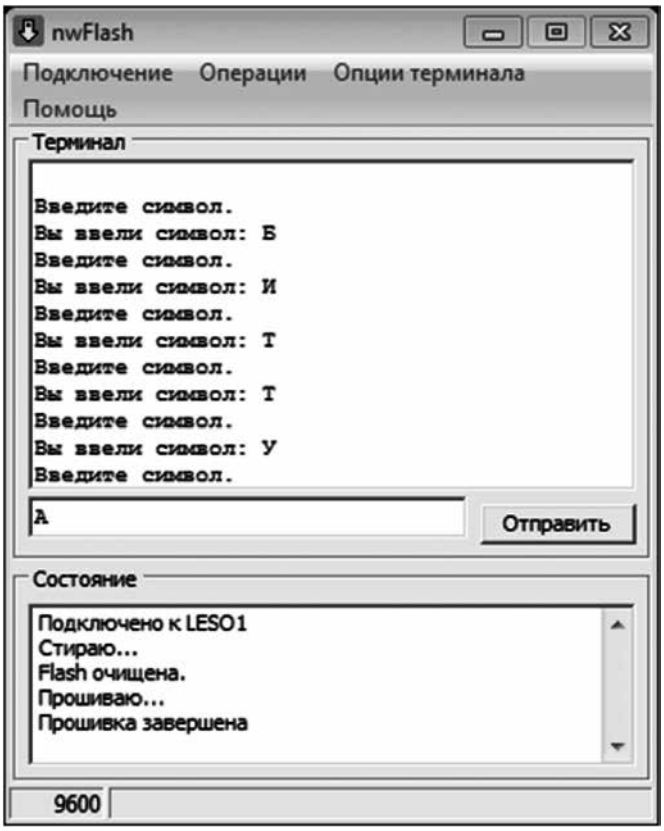


Рис. 3. Результат работы программы

В результате выполнения данной программы выводится произвольный текст.

ЛИТЕРАТУРА

1. Сонькин, М.А. Микропроцессорные системы. Применение микроконтроллеров семейства AVR для управления внешними устройствами: учебное пособие / М.А. Сонькин, Д.М. Сонькин, А.А. Шамин. — Томск: ТПУ, 2016. — 88 с. — ISBN 978-5-4387-0708-0. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/107726 (дата обращения: 15.03.2024). — Режим доступа: для авториз. пользователей.
2. Смирнов, Ю.А. Основы микроэлектроники и микропроцессорной техники: учебное пособие / Ю.А. Смирнов, С.В. Соколов, Е.В. Титов. — 2-е изд., испр. — Санкт-Петербург: Лань, 2022. — 496 с. — ISBN 978-5-8114-1379-9. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/211292 (дата обращения: 01.04.2023). — Режим доступа: для авториз. пользователей.
3. Кормилин, В.А. Вычислительная техника: учебное пособие / В.А. Кормилин. — Москва: ТУСУР, 2019. — 140 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/313487 (дата обращения: 15.03.2024). — Режим доступа: для авториз. пользователей.
4. Гаврилов, А.Н. Средства и системы управления технологическими процессами: учебное пособие / А.Н. Гаврилов, Ю.В. Пятаков. — 3-е изд., стер. — Санкт-Петербург: Лань, 2022. — 376 с. — ISBN 978-5-8114-4584-4. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/206903 (дата обращения: 01.04.2023). — Режим доступа: для авториз. пользователей.
5. Брайант, Р.Э. Компьютерные системы. Архитектура и программирование / Р.Э. Брайант, Д.Р. О'Халларон ; перевод с английского А. Н. Киселева. — 3-е изд. — Москва: ДМК Пресс, 2022. — 994 с. — ISBN 978-5-97060-492-2. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/314912 (дата обращения: 15.03.2024). — Режим доступа: для авториз. пользователей.
6. Федотов, А.В. Компьютерное управление в производственных системах: учебное пособие для вузов / А.В. Федотов, В.Г. Хомченко. — 2-е изд., стер. — Санкт-Петербург: Лань, 2021. — 620 с. — ISBN 978-5-8114-8065-4. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/171424 (дата обращения: 01.04.2023). — Режим доступа: для авториз. пользователей.
7. Микушин, А.В. Программирование микропроцессорных систем на языке С-51 / А.В. Микушин. — Санкт-Петербург: Лань, 2023. — 124 с. — ISBN 978-5-507-45539-3. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/311828 (дата обращения: 15.03.2024). — Режим доступа: для авториз. пользователей.

© Щеголев Сергей Сергеевич (SSShchegolev@mephi.ru); Ефремова Татьяна Александровна (TAEfremova@mephi.ru);
Мотков Александр Геннадьевич (AGMotkov@mephi.ru)
Журнал «Современная наука: актуальные проблемы теории и практики»