Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра проектирования информационно-
компьютерных систем

В. М. Логин, И. Н. Цырельчук, О. Ч. Ролич

ИНТЕЛЛЕКТУАЛЬНЫЕ ЭЛЕКТРОННЫЕ СИСТЕМЫ БЕЗОПАСНОСТИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 1

МИКРОКОНТРОЛЛЕРЫ СЕМЕЙСТВА AVR

Рекомендовано УМО
по образованию в области информатики
и радиоэлектроники в качестве пособия
для специальности 1-39 03 01 «Электронные системы безопасности»

Минск БГУИР 2014

УДК 004.056(076.5)
ББК 32.973.202-018.2я73
Л69

Р е ц е н з е н т ы:
кафедра автоматизированных систем управления производством учреждения образования «Белорусский государственный аграрный технический университет»
(протокол №2 от 20.09.2013);

доцент кафедры робототехнических систем Белорусского национального технического университета, кандидат технических наук, доцент Р. В. Новичихин

Логин, В. М.

Л69 Интеллектуальные электронные системы безопасности : Лабораторный практикум. В 2 ч. Ч. 1: Микроконтроллеры семейства AVR : пособие / В. М. Логин, И. Н. Цырельчук, О. Ч. Ролич. – Минск : БГУИР, 2014. – 112 с. : ил.

ISBN 978-985-543-025-5 (ч. 1).

Лабораторный практикум по дисциплине «Интеллектуальные электронные системы безопасности» состоит из восьми лабораторных работ. В первой и второй рассматривается лабораторный стенд НТЦ-31.100, в третьей – использование подпрограмм и векторов прерываний, сброс и обработка прерываний. В четвертой и пятой лабораторных работах изучаются устройства динамической и матричной жидкокристаллической индикации, в шестой – способ ввода информации при помощи клавиатуры. В седьмой и восьмой лабораторных работах рассматривается организация интерфейсов приборных шин TWI и SPI, а также работа часов реального времени.

УДК 004.056(076.5)
ББК 32.973.202-018.2я73

ISBN 978-985-543-025-5 (ч. 1)
ISBN 978-985-543-024-8

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 5
ЛАБОРАТОРНАЯ РАБОТА №1. ИССЛЕДОВАНИЕ РАБОТЫ УЧЕБНОГО СТЕНДА НТЦ-31.100 6
1.1. Микроконтроллеры семейства AVR 6
1.2. Учебный стенд НТЦ -31.100 7
1.3. Контрольные вопросы 10
ЛАБОРАТОРНАЯ РАБОТА №2. ИЗУЧЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЛАБОРАТОРНОГО СТЕНДА И СИСТЕМЫ КОМАНД МИКРОКОНТРОЛЛЕРА СЕМЕЙСТВА AVR 11
2.1. Общие сведения о системе команд микроконтроллеров семейства AVR Mega 11
2.2. Интегрированная среда AVR Studio 11
2.3. Директивы в Assembler 14
2.4. Контрольные вопросы 19
2.5. Варианты заданий 19
ЛАБОРАТОРНАЯ РАБОТА №3. ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ И ВЕКТОРОВ ПРЕРЫВАНИЙ, СБРОС И ОБРАБОТКА ПРЕРЫВАНИЙ 21
3.1. Подпрограммы 21
3.2. Вектора сброса и прерываний 23
3.3. Время реакции на прерывание 26
3.4. Пример написания программы 26
3.5. Контрольные вопросы 29
3.6. Варианты заданий 29
ЛАБОРАТОРНАЯ РАБОТА №4. ИССЛЕДОВАНИЕ УСТРОЙСТВА ДИНАМИЧЕСКОЙ ИНДИКАЦИИ 31
4.1. Устройства цифровой индикации 31
4.2. Электрическая принципиальная схема 33
4.3. Пример программы «бегущая» строка 34
4.4. Контрольные вопросы 36
4.5. Варианты заданий 36
ЛАБОРАТОРНАЯ РАБОТА №5. ИССЛЕДОВАНИЕ УСТРОЙСТВА МАТРИЧНОЙ ЖИДКОКРИСТАЛЛИЧЕСКОЙ ИНДИКАЦИИ 38
5.1. Теоретические основы и принципы работы ЖКИ 38
5.2. Рефлективные индикаторы 40
5.3. Трансмиссивные индикаторы 41
5.4. Трансрефлективные индикаторы 41
5.5. Сегменты ЖКИ 41
5.6. Жидкокристаллический индикатор SC-1602 BULT 42
5.7. Электрическая принципиальная схема 47
5.8. Контрольные вопросы 48
5.9. Варианты заданий 48
5.10. Пример выполнения лабораторной работы 49

ЛАБОРАТОРНАЯ РАБОТА №6. ИССЛЕДОВАНИЕ ВВОДА ИНФОРМАЦИИ ПРИ ПОМОЩИ КЛАВИАТУРЫ 55
6.1. Матричная организация клавиатур 55
6.2. Контрольные вопросы 56
6.3. Варианты заданий 56
6.4. Пример выполнения лабораторной работы 57
ЛАБОРАТОРНАЯ РАБОТА №7. ИССЛЕДОВАНИЕ ИНТЕРФЕЙСА ПРИБОРНОЙ ШИНЫ TWI И РАБОТЫ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ 63
7.1. Двухпроводной последовательный интерфейс TWI 63
7.2. Формат посылки и передаваемых данных 64
7.3. Модуль TWI 66
7.4. Регистры TWI 69
7.5. Часы реального времени 73
7.6. Электрическая принципиальная схема 75
7.7. Контрольные вопросы 75
7.8. Варианты заданий 75
7.9. Пример выполнения лабораторной работы 76
ЛАБОРАТОРНАЯ РАБОТА №8. ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ И РАБОТЫ ПОСЛЕДОВАТЕЛЬНОГО ПЕРИФЕРИЙНОГО ИНТЕРФЕЙСА SPI 85
8.1. Приборная шина SPI 85
8.2. Реализация интерфейса SPI в микроконтроллерах ATMega 88
8.3. Микросхема Flash-памяти (DataFlash) AT45DB041B 92
8.4. Электрическая принципиальная схема к лабораторной работе 97
8.5. Контрольные вопросы 98
8.6. Варианты заданий 98
8.7. Пример выполнения лабораторной работы 99
ПРИЛОЖЕНИЕ 1. Основные сведения о командах микроконтроллеров семейства AVR Mega 105
ПРИЛОЖЕНИЕ 2. Схема электрическая принципиальная к лабораторной работе №4 111
ЛИТЕРАТУРА 112

ВВЕДЕНИЕ

В настоящее время возросла актуальность задач по обеспечению безопасности людей, защите жилых помещений, личных автомобилей, банков, магазинов, транспорта, производственных помещений, территорий от возможных угроз: проникновения посторонних лиц, преднамеренных действий со стороны злоумышленников, действий естественных сил природы и т. п. Для решения этих задач не обойтись без интеллектуальных электронных систем безопасности (ИЭСБ).

Современные ИЭСБ представляют взаимодействующие между собой аппаратно-программные комплексы, выполняющие функции по распознаванию угроз, обработке по заранее разработанным программам сигналов об угрозах, и соответствующее реагирование на угрозы с целью их ликвидации или снижения ущерба организациям, предприятиям или физическим лицам. Специфика современных ИЭСБ состоит в том, что в своем составе кроме традиционных электронных технических частей, таких, как датчики и исполнительные устройства, они содержат компьютерные подсистемы или программируемые микропроцессорные устройства, а в качестве каналов связи между составными частями ИЭСБ используются беспроводные каналы, в том числе спутниковые и каналы мобильной радиосвязи.

Данный лабораторный практикум состоит из двух частей. В 1-й части лабораторного практикума рассматриваются системы на базе программируемых микроконтроллеров ATMega128 семейства AVR и организация интерфейсов приборной шины TWI и SPI. Во 2-й части лабораторного практикума будут рассмотрены беспроводные каналы связи между составными частями ИЭСБ, а именно спутниковая и мобильная радиосвязь.

Суть лабораторных работ сводится к практическому получению навыков работы с учебным стендом НТЦ-31.100, исходя из цели каждой работы, а также написания программного кода. Персонализация заданий к каждой лабораторной работе осуществляется посредством выдачи преподавателем каждому студенту индивидуального задания согласно вариантам заданий.

Выполнение лабораторных работ предполагает домашнюю подготовку, включающую изучение соответствующего теоретического материала курса, знакомство с инструкцией, прилагаемой к учебному стенду, изучение методики проведения лабораторных работ.

Результаты выполнения и подготовленные отчеты по каждой лабораторной работе индивидуально предъявляются студентом преподавателю и защищаются с привлечением необходимого теоретического материала из данного лабораторного практикума и лекционного курса.

Лабораторный практикум составлен так, что совершенствование прикладных учебных программ не вызывает необходимости внесения изменений в его текст.

ЛАБОРАТОРНАЯ РАБОТА №1
ИССЛЕДОВАНИЕ РАБОТЫ УЧЕБНОГО СТЕНДА НТЦ-31.100

Цель работы. Изучить состав, функциональную схему стенда, ознакомиться со структурой и принципом работы микроконтроллера, изучить порядок работы стенда.

В ходе выполнения работы необходимо изучить:

– функциональную схему стенда;

– порядок работы со стендом;

– расположение рабочих органов и разъемов подключения стенда.

Теоретическая часть

1.1. Микроконтроллеры семейства AVR

В микропроцессорной технике выделился самостоятельный класс интегральных схем – микроконтроллеры, которые предназначены для встраивания в приборы различного назначения. От класса однокристальных микропроцессоров их отличает наличие встроенной памяти, развитые средства взаимодействия с внешними устройствами. Микроконтроллеры семейства AVR являются 8-разрядными микроконтроллерами с RISK-архитектурой и имеют единую базовую структуру (рис. 1.1).

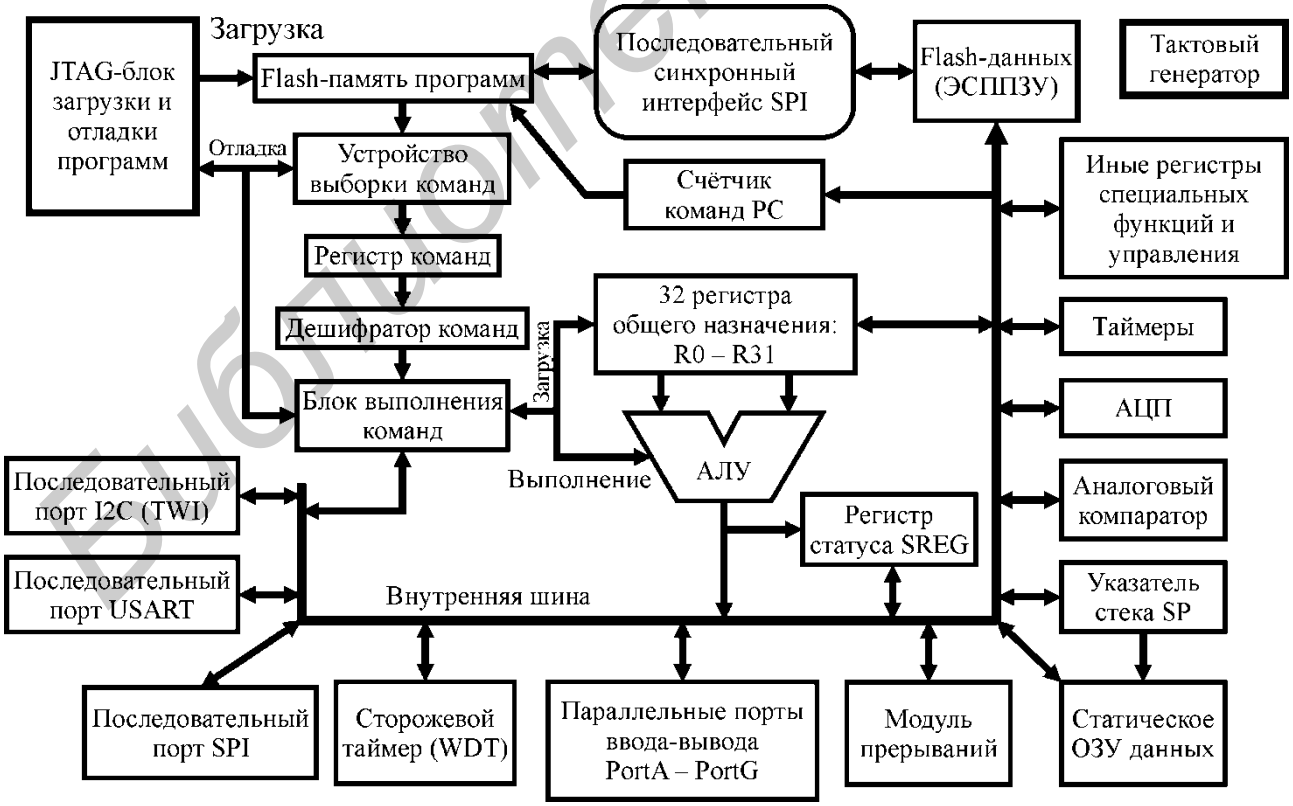


Рис. 1.1. Структурная схема микроконтроллеров семейства AVR

Обобщенно структуру микроконтроллера можно представить в виде ядра, состоящего из арифметическо-логического устройства, 32 регистров общего назначения, статического оперативного запоминающего устройства (ОЗУ), Flash-памяти программ, энергонезависимой памяти данных (ЭСППЗУ), логики управления и комплекса периферийных устройств, в который входят параллельные порты ввода-вывода, последовательные порты ввода-вывода, многофункциональные таймеры-счетчики, блок прерываний, сторожевой таймер, аналоговый компаратор и другие устройства.

1.2. Учебный стенд НТЦ -31.100

1.2.1. Структура учебного стенда НТЦ -31.100

Учебный стенд НТЦ-31.100 состоит из следующих структурных элементов (рис. 1.2).

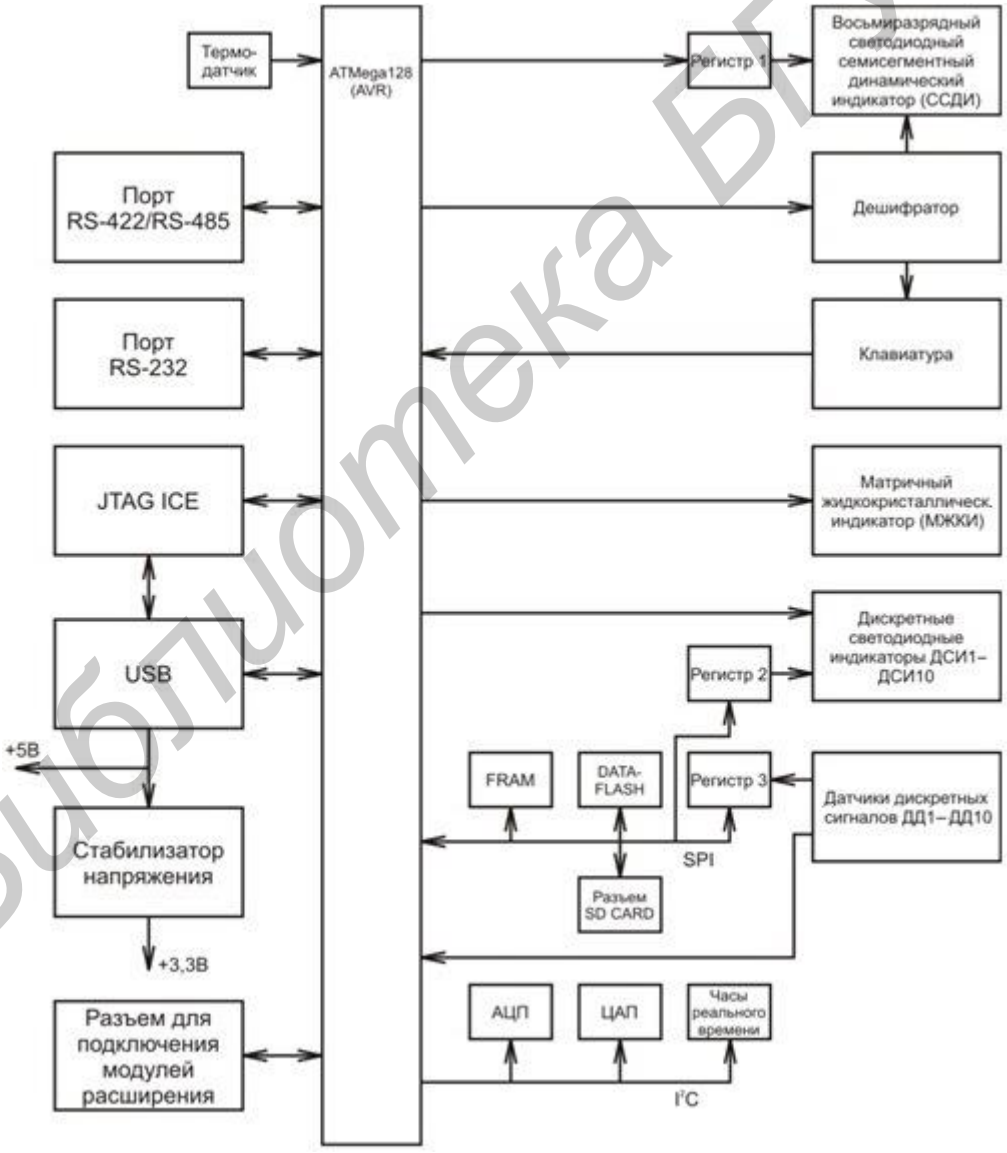


Рис. 1.2. Функциональная схема учебного стенда СУ-МК НТЦ-31.100

Стенд построен на базе микроконтроллера ATMega128 семейства AVR. Для исследования вывода дискретных сигналов используются дискретные светодиодные индикаторы ДСИ1–ДСИ10. Для исследования применения динамической 7-сегментной индикации используется 8-разрядный светодиодный 7-сегментный динамический индикатор (ССДИ), управление которым осуществляется через параллельный регистр 1 и дешифратор. Для исследования ввода дискретных сигналов используются датчики дискретных сигналов ДД1–ДД10. К микроконтроллеру подключен матричный жидкокристаллический индикатор МЖКИ и 12-кнопочная клавиатура.

В стенде организована шина I2C, по которой к микроконтроллеру подключены: АЦП, ЦАП (цифровой резистор) и часы реального времени с электрически стираемым ППЗУ.

В стенде организована шина SPI, по которой реализована связь микроконтроллера и энергонезависимого ОЗУ (FRAM), электрически стираемого ПЗУ (DATAFLASH). По шине SPI к микроконтроллеру также подключены разъем для подключения FLASH-карт памяти (SD, MMC) и два последовательных регистра, через регистр 3 осуществляется ввод в микроконтроллер сигналов с датчиков дискретных сигналов ДД3–ДД10, а через регистр 2 – вывод сигналов с микроконтроллера на дискретные светодиодные индикаторы ДСИ3–ДСИ10. Для исследования интерфейса 1-wire стенд оснащен термодатчиком, подключенным к микроконтроллеру.

Для связи с другими устройствами стенд оснащен последовательными портами RS232, RS422/RS485. Стенд оснащен разъемом подключения внешних модулей расширения.

Отладка программ, загружаемых в микроконтроллер, осуществляется посредством встроенного в микроконтроллер модуля внутрисхемной отладки, который взаимодействует с программной интегрированной средой разработки программ для микроконтроллеров на персональном компьютере через модуль JTAG ICE, входящий в состав стенда. Подключение модуля JTAG ICE к персональному компьютеру осуществляется через USB порт.

Все устройства, входящие в состав стенда, кроме JTAG ICE, являются программно-доступными.

1.2.2. Органы управления стенда НТЦ-31.100

На передней панели учебного стенда (рис. 1.3) расположены:

– дискретные светодиодные индикаторы (ДСИ1–ДСИ10) 1;
– светодиодный 7-сегментный динамический индикатор (ССДИ) 2;
– матричный жидкокристаллический индикатор (МЖКИ) 3;
– линейка светодиодиодных индикаторов (ЛСИ) 4;
– имитаторы аналогового сигнала на входах АЦП АД1–АД3 5;
– датчики дискретных сигналов (ДД1–10 6);
– кнопка сброса 7;
– 12-кнопочная клавиатура (КЛ) 8.

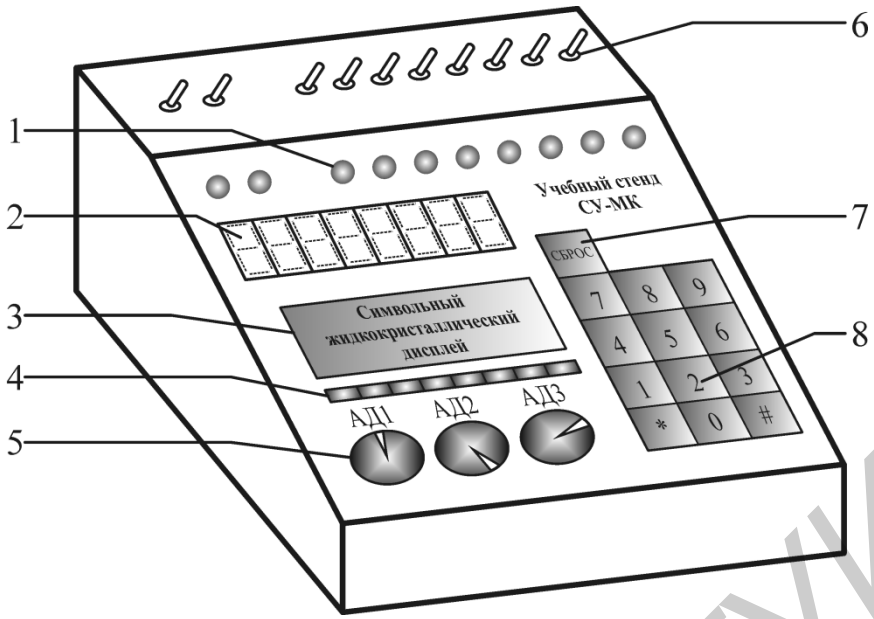


Рис. 1.3. Внешний вид передней панели учебного стенда

На задней панели лабораторного стенда (рис. 1.4) расположены:

– выключатель «Питание» 1;

– разъем «RS232» для подключения стенда к ПК 2;

– разъем «RS422/RS485» для подключения внешних устройств 3;

– разъем «USB» для подключения стенда к ПК 4;

– разъем «Внешние устройства» для подключения дополнительных внешних устройств 5;

– разъем «SD CARD» для подключения к стенду Flash-карт памяти (SD, MMC) 6.

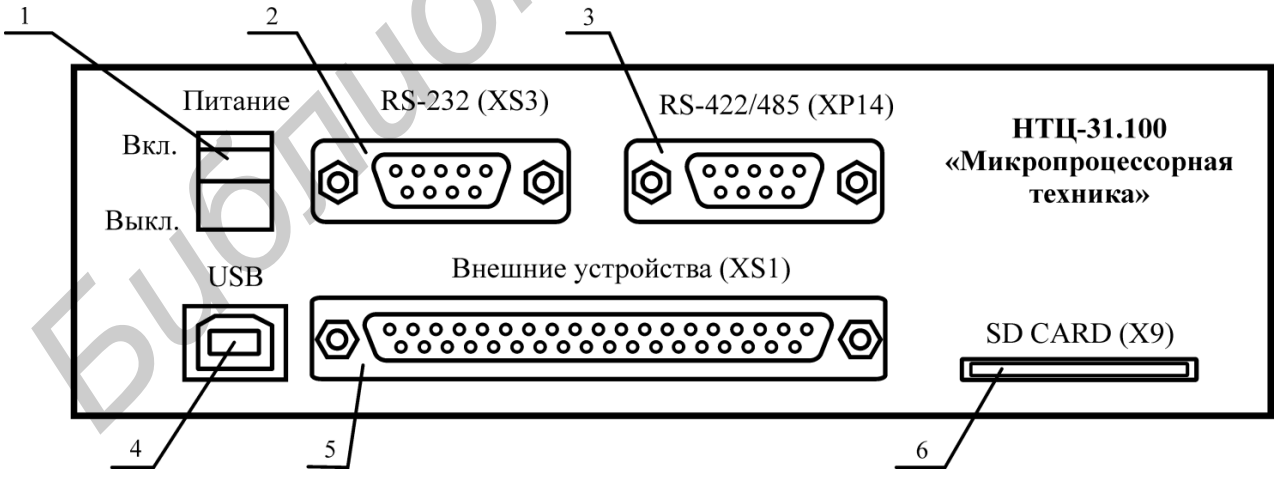


Рис. 1.4. Вид задней панели учебного стенда

Практическая часть

1.2.3. Порядок работы учебного стенда НТЦ-31.100

1. Подключить стенд к персональному компьютеру. Для этого необходимо подключить прилагаемый к стенду кабель к разъему «USB» стенда и к одному из разъемов USB персонального компьютера.

2. Выключатель «Питание» на задней панели стенда установить в положение «Вкл.».

3. Загрузить на персональном компьютере интегрированную среду разработки программ для микроконтроллеров семейства AVR «IAR Embedded Workbench for Atmel AVR kickstart».

4. Открыть рабочую среду с проектом.

5. Скомпилировать проект.

6. Запустить отладчик скомпилированной программы, при этом скомпилированная программа автоматически загрузится во FLASH-память программ микроконтроллера стенда. После загрузки отладчик находится в состоянии пошаговой отладки, в точке старта отлаживаемой программы.

7. Выполнить все необходимые действия, связанные с отладкой программы.

8. Закрыть отладчик.

9. По окончании работы со стендом выключатель «Питание» установить в положение «Выкл.».

1.3. Контрольные вопросы

1. Перечислите характерные черты архитектуры однокристальных микроконтроллеров.

2. Какие структурные элементы входят в состав микроконтроллеров семейства AVR?

3. Каковы основные структурные элементы учебного стенда НТЦ-31.100?

4. Опишите органы управления лабораторного стенда.

5. Каков порядок работы со стендом?

Содержание отчета по лабораторной работе: цель работы, задание, ход выполнения работы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №2
ИЗУЧЕНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЛАБОРАТОРНОГО СТЕНДА И СИСТЕМЫ КОМАНД МИКРОКОНТРОЛЛЕРА СЕМЕЙСТВА AVR

Цель работы. Изучить состав и особенности выполнения команд микроконтроллера, программное обеспечение лабораторного стенда, освоить технику программной симуляции работы микроконтроллера семейства AVR на примере построения элементарных конструкций.

В ходе выполнения работы необходимо:

– составить фрагменты программ для выполнения заданных операций;

– набрать тексты программ в редакторе программы AVR Studio, откомпилировать и загрузить в отладчик;

– исследовать выполнение команд, входящих в состав программ;

– составить отчет о выполнении работы.

Теоретическая часть

2.1. Общие сведения о системе команд микроконтроллеров семейства AVR Mega

Система команд микроконтроллеров семейства AVR Mega весьма развита и насчитывает в различных моделях от 130 до 135 различных команд, которые по функциональному признаку могут быть разделены на следующие группы:

– арифметические команды и команды сдвига;

– логические команды;

– команды передачи управления;

– команды пересылки данных;

– команды битовых операций;

– команды управления системой.

Большинство команд имеют формат в одно 16-разрядное слово. Исключение составляют команды, у которых одним из операндов является 16-битный адрес. Длительность выполнения команд составляет от одного до четырех тактов задающего генератора (зависит от типа команды).

В прил. 1 приведены основные сведения о командах, такие, как мнемоническое обозначение команды, ее описание, число тактов, необходимых для ее выполнения, а также флаги регистра состояния, на которые воздействует эта команда.

2.2. Интегрированная среда AVR Studio

Для работы с данной средой необходимо запустить ярлык AVR Studio 4. В меню выбираем Project->Project Wizard. Откроется диалог создания нового или открытия существующего проекта (рис. 2.1). Нажимаем кнопку New

Project. Теперь нам предлагают выбрать тип проекта. Так как мы будем писать прошивку на языке Assembler, выбираем тип проекта AVR GCC. Здесь же выбираем имя проекта, имя файла, содержащего код (надо поставить галочку "Create initial file", если не стоит), и путь, где проект будет сохранен. Если установить флаг "Create folder", в выбранном каталоге будет создан подкаталог с именем, совпадающим с именем проекта.

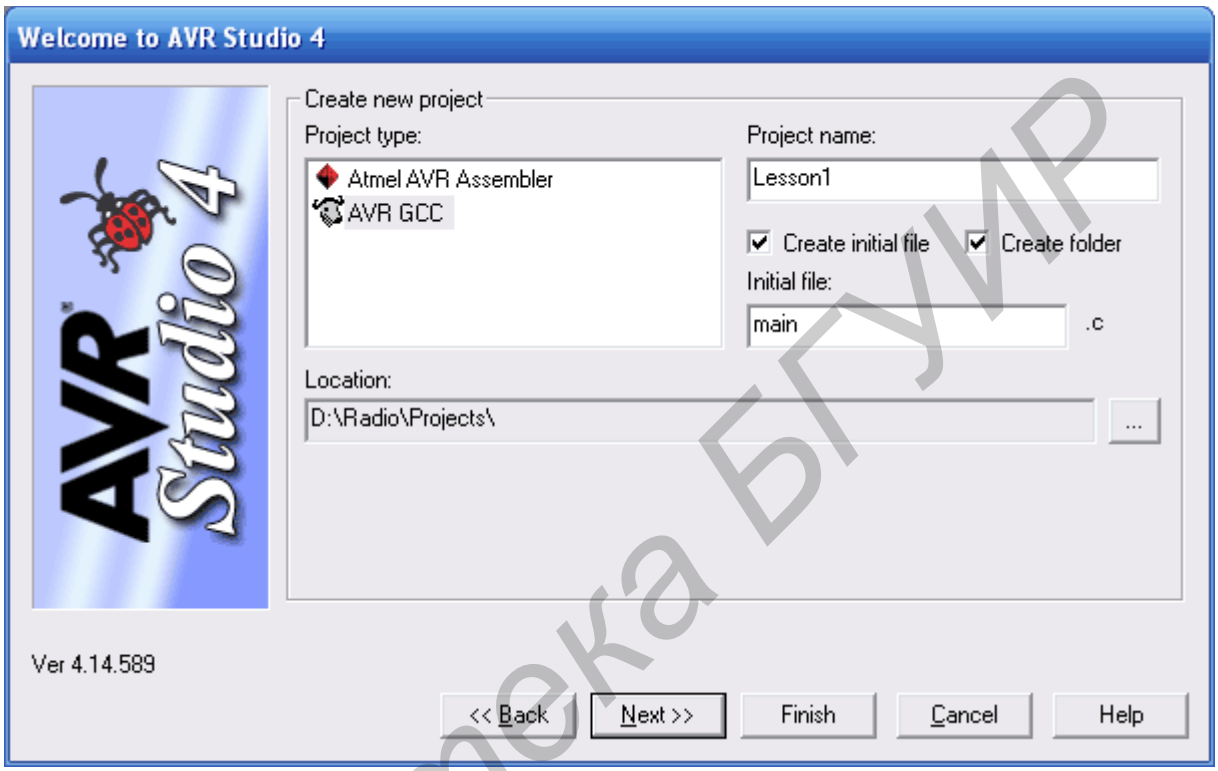


Рис. 2.1. Диалог создания нового или открытия существующего проекта

Далее мастер предлагает выбрать отладчик (рис. 2.2). Если у вас нет аппаратного отладчика, выбирайте AVR Simulator. Это эмулятор микроконтроллера, позволяющий отладить вашу прошивку, отложив сборку устройства. Кроме того, необходимо выбрать тип микроконтроллера, под который пишется прошивка. Пример введенных данных на этом этапе создания проекта представлен на рис. 2.2.

После написания кода для сборки прошивки нажимаем клавишу F7 или выбираем пункт меню Buid->Build. Программа должна написать в окне Build о том, что сборка прошла успешно. Теперь можно запустить отладку прошивки. Для этого нажмите комбинацию клавиш Ctrl+Alt+Shift+F5 или выберите пункт меню Debug->Startdebugging. Выполнение отладки сразу же остановится на первой команде вашей программы.

Ниже, в табл. 2.1 перечислены основные команды, которые могут понадобиться при отладке программы.

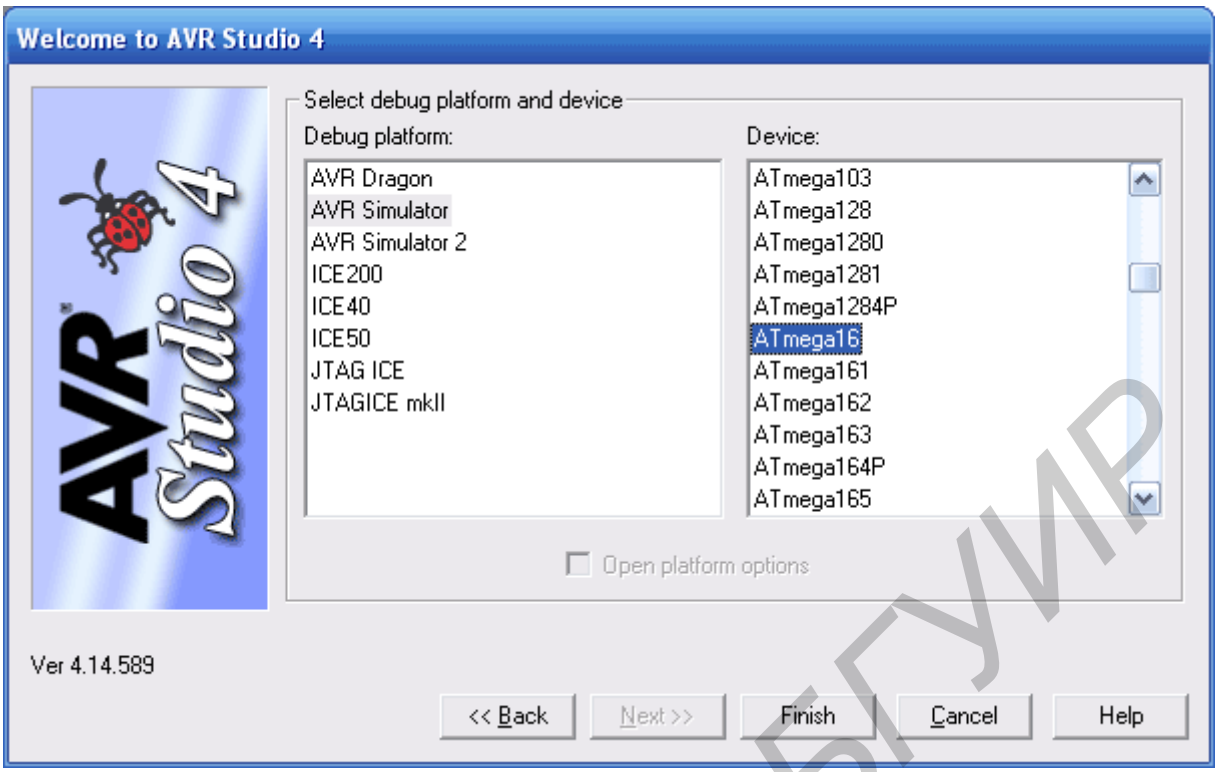


Рис. 2.2. Окно выбора отладчика

Таблица 2.1

Основные команды отладчика

Горячая клавиша	Команда	Описание команды
Ctrl+Alt+Shift+F5	StartDebugging	Запуск процесса отладки
Ctrl+Shift+F5	StopDebugging	Прерывание процесса отладки
F10	StepOver	Сделать шаг без захода в функцию
F11	StepInto	Сделать шаг с заходом в функцию
Shift+F11	StepOut	Выполнить программу до выхода из текущей функции
Ctrl+F10	RuntoCursor	Выполнить программу до той строки кода, где установлен курсор

2.2.1. Основные элементы главного окна

I/O View

Окно периферии, которое находится справа. Обратите внимание на заголовок окна I/O View – там есть строка поиска и ряд рабочих кнопок.

View

Каждый регистр можно развернуть. Каждый бит подписан и во всплывающей подсказке написано его назначение. Также настоятельно рекомендуется просмотреть контекстное меню в этом блоке и выставить галку Show Bitnumbers. Черный квадратик – бит есть, белый – бита нет. Там же указано

значение в байте. В процессе отладки каждый бит можно вручную принудительно выставить или сбросить, сэмулировав, например, нажатие кнопки.

Processor

Слева есть окно Processor, в нем указана вся подноготная ядра в текущий момент.

Memory

Окно памяти. В нем можно в списке выбирать любое адресное пространство.

Watch

Это лист просмотра. В нем можно на любой адрес, переменную или любой регистр навести курсор просмотра, где всегда будет отображаться его содержимое. Это бывает удобно при отладке программы, чтобы не просматривать весь код. Также содержимое регистров/переменных подсвечивается в подсказке при наведении мыши.

2.2.2. Отладка в AVR Studio

Выставление частоты процессора в AVR Studio

Нужный пункт меню появится только при запущенной симуляции. После выбираем в меню Debug -> AVR Simulations Options.

BreakPoint

Это точка остановки. То есть если какое-то условие совпадет, то процессор останавливается, пока вы не примете решение, что же делать дальше. Самый простой случай – это установка Breakpoint кнопкой F9, она же его и отключает. Через контекстное меню Breakpoint можно временно деактивировать ход выполнения программы. Если теперь вы нажмете на F5, то процессор будет работать, пока не дойдет до Breakpoint, где остановится, перейдя в пошаговый режим.

2.2.3. Специализированные симуляторы

Для проверки правильности воздействия прошивки на периферийные устройства удобно использовать программу Proteus. В окне эмулятора Proteus (рис. 2.3) показана эмуляция стенда с ЖКИ, матричной клавиатурой, светодиодами, подключенными к PortG, и светодиодным 7-сегментным дисплеем. Данный инструмент позволяет создавать различные электронные схемы для симуляции.

2.3. Директивы в Assembler

1.Исходные коды

Компилятор работает с исходными файлами, содержащими инструкции, метки и директивы. Инструкции и директивы, как правило, имеют один или несколько операндов.

Строка кода не должна быть длиннее 120 символов.

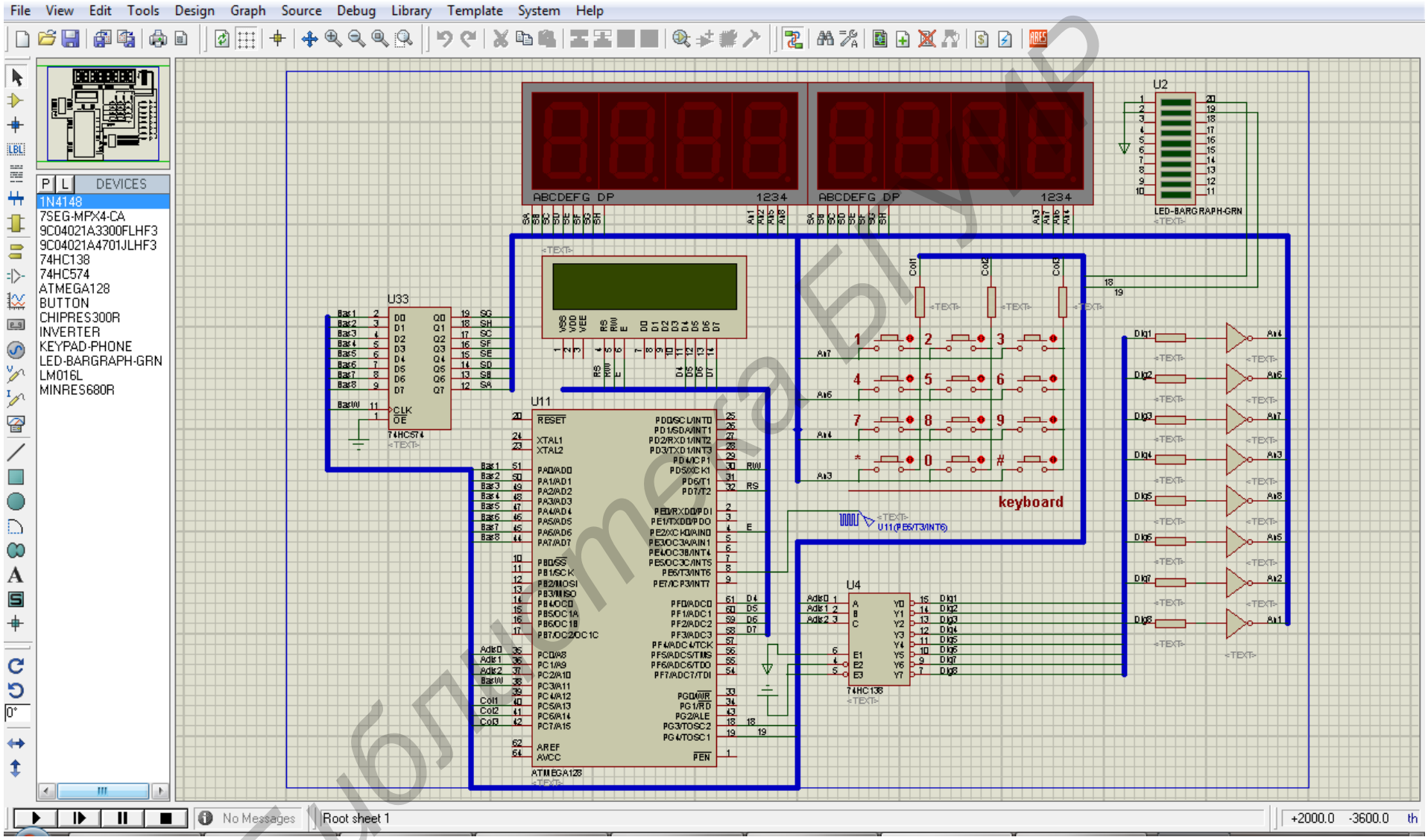


Рис. 2.3. Окно эмулятора Proteus

Любая строка может начинаться с метки, которая является набором символов, заканчивающимся двоеточием. Метки используются для указания места, в которое передается управление при переходах, а также для задания имен переменных.

Входная строка может иметь одну из четырех форм:

- *[метка:] директива [операнды] [Комментарий]*
- *[метка:] инструкция [операнды] [Комментарий]*
- *Комментарий*
- *Пустая строка*

Комментарий имеет следующую форму:
; [Текст]

Позиции в квадратных скобках необязательны. Текст после точки с запятой (;) и до конца строки игнорируется компилятором. Метки, инструкции и директивы более детально описываются ниже.

```
label:
.EQU var1=100          ; Устанавливает var1 равным 100 (директива)
.EQU var2=200          ; Устанавливает var2 равным 200
                       ; Строка с одним только комментарием
```

Компилятор не требует, чтобы метки, директивы, комментарии или инструкции находились в определенной колонке строки.

2. Резервирование байтов в ОЗУ

Директива BYTE резервирует байты в ОЗУ. Если вы хотите иметь возможность ссылаться на выделенную область памяти, то директива BYTE должна быть предварена меткой. Директива принимает один обязательный параметр, который указывает количество выделяемых байт. Эта директива может использоваться только в сегменте данных (см. директивы CSEG и DSEG). Выделенные байты не инициализируются.

Синтаксис:
МЕТКА: .BYTE выражение

Пример:

```
.DSEG
var1: .BYTE 1              ; Резервирует 1 байт для var1
table: .BYTE tab_size      ; Резервирует tab_size байт
.CSEG
ldi r30,low(var1)          ; Загружает младший байт регистра Z
ldi r31,high(var1)         ; Загружает старший байт регистра Z
ld r1,Z                    ; Загружает VAR1 в регистр 1
```

3. Программный сегмент

Директива CSEG определяет начало программного сегмента. Исходный файл может состоять из нескольких программных сегментов, которые объединяются в один программный сегмент при компиляции. Программный сегмент является сегментом по умолчанию. Программные сегменты имеют свои собственные счетчики положения, которые считают не побайтно, а пословно. Директива ORG может быть использована для размещения кода и констант в необходимом месте сегмента. Директива CSEG не имеет параметров.

Синтаксис:
.CSEG

Пример:
```
.DSEG
vartab: .BYTE 4          ; Резервирует 4 байта в ОЗУ
.CSEG                    ; Начало кодового сегмента
const: .DW 2             ; Разместить константу 0x0002 в памяти
                         ; программ
mov r1,r0                ; Выполнить действия
```

4. Назначение регистру символического имени

Директива DEF позволяет ссылаться на регистр через некоторое символическое имя. Назначенное имя может использоваться во всей нижеследующей части программы для обращений к данному регистру. Регистр может иметь несколько различных имен. Символическое имя может быть переназначено позднее в программе.

Синтаксис:
.DEF Символическое_имя = Регистр

Пример:
```
.DEF temp=R16
.DEF ior=R0
.CSEG
ldi temp,0xf0            ; Загрузить 0xf0 в регистр temp (R16)
in ior,0x3f              ; Прочитать SREG в регистр ior (R0)
eortemp,ior              ; Регистры temp и ior складываются по
                         ; исключающему ИЛИ
```

5. Сегмент данных

Директива DSEG определяет начало сегмента данных. Исходный файл может состоять из нескольких сегментов данных, которые объединяются в один сегмент при компиляции. Сегмент данных обычно состоит только из директив BYTE и меток. Сегменты данных имеют свои собственные побайтные счетчики

положения. Директива ORG может быть использована для размещения переменных в необходимом месте ОЗУ. Директива не имеет параметров.

Синтаксис:
.DSEG

Пример:
```
.DSEG                       ; Начало сегмента данных
var1: .BYTE 1               ; Зарезервировать 1 байт для var1
table: .BYTE tab_size       ; Зарезервировать tab_size байт.
.CSEG
ldi r30,low(var1)           ; Загрузить младший байт регистра Z
ldi r31,high(var1)          ; Загрузить старший байт регистра Z
ld r1,Z                     ; Загрузить var1 в регистр r1
```

6. Установка постоянного выражения

Директива EQU присваивает метке значение. Эта метка может позднее использоваться в выражениях. Метка, которой присвоено значение данной директивой, не может быть переназначена и ее значение не может быть изменено.

Синтаксис:
.EQU метка = выражение

Пример:
```
.EQU io_offset = 0x23
.EQU porta  = io_offset + 2
.CSEG                       ; Начало сегмента данных
clr r2                      ; Очистить регистр r2
out porta,r2                ; Записать в порт A
```

7. Вложение файла

Встретив директиву INCLUDE, компилятор открывает указанный в ней файл, компилирует его, пока файл не закончится или не встретится директива EXIT, после этого продолжает компиляцию начального файла со строки, следующей за директивой INCLUDE. Вложенный файл может также содержать директивы INCLUDE.

Синтаксис:
.INCLUDE "имя_файла"

Пример:
```
                            ; Файл iodefs.asm:
.EQU sreg = 0x3f            ; Регистр статуса
.EQU sphigh = 0x3e          ; Старший байт указателя стека
```

```
.EQU splow = 0x3d          ; Младший байт указателя стека

                           ; Файл incdemo.asm
.INCLUDE iodefs.asm        ; Вложить определения портов
in r0,sreg                 ; Прочитать регистр статуса
```

Практическая часть

2.4. Контрольные вопросы

1. Сколько команд включает система команд микроконтроллера?
2. На какие группы разделены эти команды?
3. Как длительность машинного цикла микроконтроллера соотносится с его тактовой частотой?
4. С какими типами данных может оперировать микроконтроллер?
5. Укажите назначение флагов регистра SREG.
6. Может ли порт одновременно являться источником операнда и приемником результата операции?
7. Какие способы адресации используются в микроконтроллере?
8. Приведите примеры логических и арифметических команд.
9. Как инвертировать отдельные биты портов?
10. Какие переходы возможны в командах управления?
11. Поясните отличия длинного, абсолютного и относительного переходов в программах.
12. Как организовать процедуру ожидания с помощью одной команды?
13. Какие команды используются при организации подпрограмм?

2.5. Варианты заданий

1. Загрузить в регистр R16 число 15, сложить его с 25 и результат поместить на вершину стека. Поместить по адресу 020h внутренней памяти данных младшую десятичную цифру результата, а по адресу 021h – старшую.
2. Найти разницу чисел 4836 и 232. Младший байт результата поделить на 2. Поместить по адресу 025h внутренней памяти данных младший байт, а по адресу 030h старший байт результата.
3. Найти адрес ячейки памяти данных путем перемножения двух чисел 0Ch и 0Eh. В эту ячейку записать результат логической операции «исключающее ИЛИ» между текущим содержимым регистра R0 и числа 09h.
4. Найти частное чисел 236 и 59. Результат умножить на 23, используя операции сдвига. По вычисленному таким образом адресу ячейки внутренней памяти данных разместить результат двойного декремента полученного числа.
5. В младшую тетраду порта PORTB вывести число десятков от числа 044h. Старшую тетраду необходимо оставить без изменений.

6. Загрузить регистр R17 числом 023h. Найти сумму R17+SREG. В ячейку внутренней памяти данных, расположенную по вычисленному таким образом адресу, загрузить число десятичных единиц результата сложения.

7. Найти сумму чисел 9701 и 32. Младший байт результата умножить на 4. Поместить старший байт в порт PORTC. Сбросить младшую тетраду байта порта.

8. Вычислить значение выражения (81+64)·(112–25) OR 10011010b, сохраняя промежуточные результаты в стеке.

9. Загрузить в регистр R18 число 112, вычесть из него число 18 и результат поместить на вершину стека. Поместить по адресу 020h внутренней памяти данных младшую десятичную цифру результата, а по адресу 021h – старшую.

10. Найти разницу чисел 4801 и 209. Число десятичных единиц старшего байта результата поместить в старшую тераду порта PORTD. Младшую тетраду оставить без изменений.

11. Загрузить регистр R19 числом 0Ah. Найти произведение 8·R19. Результат разделить на 4, используя операции сдвига. В две ячейки внутренней памяти данных, начиная с ячейки, расположенной по вычисленному таким образом адресу, загрузить число 023E8h.

12. В ячейки внутренней памяти данных 128h, 129h, 12ah занести число сотен, десятков, единиц числа 080h.

13. Вычислить младший байт адреса ячейки внутренней памяти данных 7XXh как произведение 0A1h и 7, поместить по этому адресу значение выражения NOT (0101001b OR 74).

14. Вычислить значение выражения (72+56)·(122–15) XOR 01010011b, сохраняя промежуточные результаты в стеке.

15. Найти среднее арифметическое чисел 012h, 033h, 0Ah. Результат умножить на 22, используя операции сдвига. Младшую тераду полученного числа разместить в старшей тетраде порта P1, а старшую – в младшей.

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №3
ИСПОЛЬЗОВАНИЕ ПОДПРОГРАММ И ВЕКТОРОВ ПРЕРЫВАНИЙ, СБРОС И ОБРАБОТКА ПРЕРЫВАНИЙ

Цель работы. Ознакомиться с использованием подпрограмм, порядком обработки прерываний, разрешением и запретом прерываний, изучить размещение прерываний.

В ходе выполнения работы необходимо:

– изучить понятия прерываний и подпрограмм;

– разработать программу в соответствии с индивидуальным заданием;

– отладить программу в среде Atmel«AVRStudio»;

– загрузить программу в учебный стенд НТЦ-31.100;

– исследовать адресацию при обработке прерываний и использовании подпрограмм;

– оформить отчет по лабораторной работе.

Теоретическая часть

3.1. Подпрограммы

Когда один и тот же участок кода часто повторяется, то разумно как-то его вынести и использовать многократно. Это дает колоссальный выигрыш по объему кода и удобству программирования.

Вот, например, кусок кода, передающий в регистр UDR байты с некоторой выдержкой, выдержка делается за счет вращения бесконечного цикла:

```
.CSEG           LDI             ; Инициализация стека
                R16,Low(RAMEND)
                OUT SPL,R16
                LDI
                R16,High(RAMEND)
                OUT SPH,R16
                .equ Byte = 50  ; Загрузили значение
                .equ Delay = 20
Start:                          ; Выдали его в порт
                LDI  R16,Byte   ; Загрузили длитель-
                OUT UDR,R16     ; ность задержки
M1:             LDI  R17,Delay  ; Уменьшили на 1
                                ; Пустая операция
                DECR17          ; Длительность не
                NOP             ; равна 0?
                BRNEM1          ; Переход, если не 0
                                ; Выдали значение
                OUTUDR,R16      ; в порт
```

```
M2:     LDI  R17,Delay      ; Аналогично
        DEC  R17
        NOP
        BRNE M2
        OUT  UDR,R16

        LDI  R17,Delay
M3:     DEC  R17
        NOP
        BRNEM3
        RJMP Start          ; Зациклим программу
```

Сразу напрашивается повторяющийся участок кода вынести за скобки.

```
        LDI  R17,Delay
        DEC  R17
M2:     NOP
        BRNE M2
```

Для этих целей есть группа команд перехода к подпрограмме CALL (ICALL, RCALL, CALL) и команда возврата из подпрограммы RET.

В результате получается следующий код:

```
.CSEG   LDI                 ; Инициализация стека
        R16,Low(RAMEND)
        OUT SPL,R16
        LDI
        R16,High(RAMEND)
        OUT SPH,R16
        .equ Byte = 50      ; Загрузили значение
        .equ Delay = 20
Start:                      ; Выдали его в порт
        LDI  R16,Byte
        OUT  UDR,R16

        RCALL Wait
        OUT  UDR,R16
        RCALL Wait
        OUT  UDR,R16
        RCALL Wait
        OUT  UDR,R16        ; Зациклим программу
        RCALL Wait
Wait:   RJMP Start
```

```
M1:     LDI  R17,Delay
        DEC R17
        NOP
        BRNE M1
        RET
```

Как видно, программа значительно сократилась в размерах. Чем больше количество повторений, тем более явным становится очевидное преимущество такой замены.

Если речь идет о повторении малого участка кода, то, вероятно, более оптимальным вариантом написания кода будет использование макросов.

3.2. Векторы сброса и прерываний

Наименьшие адреса в памяти программ по умолчанию определены как векторы сброса и прерываний. Полный перечень векторов приведен в табл. 3.1. В перечне также определяется уровень приоритетов различных прерываний. Меньшие адреса обладают более высоким уровнем, приоритетом.

Сброс (RESET) имеет наивысший приоритет, за ним следует INT0 запрос на внешнее прерывание по входу INT0. Векторы прерывания могут быть перемещены в начало загрузочного сектора Flash-памяти установкой бита IVSEL в регистре управления микроконтроллером (MCUCR). Вектор сброса может быть также перемещен в начало загрузочного сектора Flash-памяти путем программирования конфигурационного бита BOOTRST.

После возникновения прерывания бит I общего разрешения прерываний сбрасывается, и все прерывания запрещаются. Пользователь может программно записать логическую «1» в бит I для разрешения вложенных прерываний. В этом случае все разрешенные прерывания могут прервать текущую процедуру обработки прерываний. Бит I автоматически устанавливается после выполнения инструкции выхода из прерывания RETI.

Имеются два основных типа прерываний. Первый тип прерываний активизируется событием, которое приводит к установке флага прерываний. Для данных прерываний программный счетчик изменяется на соответствующий вектор прерывания для выполнения процедуры его обработки и затем аппаратно очищает флаг прерывания. Флаги прерывания также сбрасываются путем записи логической «1» в соответствующий разряд. Если возникает условие прерывания, но данное прерывание запрещено, то флаг устанавливается и запоминается до разрешения этого прерывания или сбрасывается программно.

Аналогично, если возникает одно и более условий прерываний при сброшенном флаге общего разрешения прерываний, то соответствующий флаг устанавливается и запоминается до возобновления работы прерываний, а затем прерывания будут выполнены в соответствии с приоритетом.

Второй тип прерываний активизируется сразу после выполнения условия прерывания. Данные прерывания необязательно имеют флаги прерываний. Ес-

ли условие прерывания исчезает до его разрешения, то данный запрос игнорируется. После выхода из прерывания AVR-микроконтроллер возвращается к выполнению основной программы и выполняет еще одну инструкцию до обслуживания любого из отложенных прерываний.

Необходимо обратить внимание, что регистр статуса автоматически не запоминается при вызове процедуры обработки прерывания и не восстанавливается при выходе из этой процедуры. Данные действия необходимо выполнить программно.

При выполнении инструкции CLI все прерывания запрещаются. Запрос на прерывание не будет отработан после выполнения инструкции CLI, даже если оно возникает одновременно с выполнением команды CLI.

Таблица 3.1

Векторы сброса и прерываний

Номер вектора	**Адрес памяти программ**	**Источник**	**Условие возникновения прерывания**
1	$0000	RESET	Внешний сброс, сброс при подаче питания, сброс при недопустимом снижении питания, сброс сторожевым таймером и сброс через JTAG-интерфейс
2	$0002	INT0	Запрос на внешнее прерывание 0
3	$0004	INT1	Запрос на внешнее прерывание 1
4	$0006	INT2	Запрос на внешнее прерывание 2
5	$0008	INT3	Запрос на внешнее прерывание 3
6	$000A	INT4	Запрос на внешнее прерывание 4
7	$000C	INT5	Запрос на внешнее прерывание 5
8	$000E	INT6	Запрос на внешнее прерывание 6
9	$0010	INT7	Запрос на внешнее прерывание 7
10	$0012	TIMER2 COMP	Срабатывание компаратора таймера-счетчика 2
11	$0014	TIMER2 OVF	Переполнение таймера-счетчика 2
12	$0016	TIMER1 CAPT	Захват фронта таймером-счетчиком 1
13	$0018	TIMER1 COMPA	Срабатывание компаратора A таймера-счетчика 1
14	$001A	TIMER1 COMPB	Срабатывание компаратора B таймера-счетчика 1
15	$001C	TIMER1 OVF	Переполнение таймера-счетчика 1
16	$001E	TIMER0 COMP	Срабатывание компаратора таймера-счетчика 0

Окончание табл. 3.1

Номер вектора	**Адрес памяти программ**	**Источник**	**Условие возникновения прерывания**
17	$0020	TIMER0 OVF	Переполнение таймера-счетчика 0
18	$0022	SPI, STC	Завершение последовательной передачи интерфейсом SPI
19	$0024	USART0, RX	Завершение приема УСАПП 0
20	$0026	USART0, UDRE	Регистр данных УСАПП 0 свободен
21	$0028	USART0, TX	Завершение передачи УСАПП 0
22	$002A	ADC	Завершение преобразования АЦП
23	$002C	EE READY	Готовность ЭСППЗУ
24	$002E	ANALOG COMP	Аналоговый компаратор
25	$0030	TIMER1 COMPC	Срабатывание компаратора С таймера-счетчика 1
26	$0032	TIMER3 CAPT	Захват фронта таймером-счетчиком 3
27	$0034	TIMER3 COMPA	Срабатывание компаратора А таймера-счетчика 3
28	$0036	TIMER3 COMPB	Срабатывание компаратора В таймера-счетчика 3
29	$0038	TIMER3 COMPC	Срабатывание компаратора С таймера-счетчика 3
30	$003A	TIMER3 OVF	Переполнение таймера-счетчика 3
31	$003C	USART1, RX	Завершение приема УСАПП 1
32	$003E	USART1, UDRE	Регистр данных УСАПП 1 свободен
33	$0040	USART1, TX	Завершение передачи УСАПП 1
34	$0042	TWI	2-проводной последовательный интерфейс
35	$0044	SPM READY	Готовность записи в память программ

3.3. Время реакции на прерывание

Реакция на отработку запроса на прерывание длится минимум 4 машинных цикла. По истечении этого времени программа продолжает свое выполнение с вектора соответствующего прерывания. В течение 4 машинных циклов состояние программного счетчика помещается в стек. Как правило, по адресу вектора прерываний хранится команда перехода на процедуру обработки прерываний, а на данный переход затрачивается еще 3 машинных цикла. Если запрос на прерывание возникает в процессе исполнения инструкции, требующей более 1 машинного цикла на выполнение, то прерывание будет обработано только после выполнения этой инструкции. Если прерывание возникает во время нахождения микроконтроллера в режиме сна, то реакция на прерывание увеличится еще на 4 цикла. Данная задержка связана со временем старта из выбранного режима сна.

Выход из процедуры обработки прерывания требует 4 машинных цикла. В течение этого времени 2-байтный программный счетчик извлекается из стека, указатель стека дважды инкрементируется и устанавливается бит I в регистре статуса SREG.

3.4. Пример написания программы

Рассмотрим программу, выполняющую запись единиц в регистры R16, R17, R18 при падающем фронте на входе PE6 (INT6). Регистры, используемые в программе, и принципы задания условий генерации запроса на прерывание представлены в табл. 3.2–3.5.

Таблица 3.2

Регистр маски прерывания EIMSK

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
Чтение/запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Если в биты INT0–INT7 и в бит I регистра статуса SREG записать логическую «1», то разрешается работа внешнего прерывания по соответствующему выводу. Биты выбора условия генерации прерывания в регистрах управления внешними прерываниями EICRA и EICRB определяют, по какому условию генерируется прерывание: по нарастающему фронту, по падающему фронту или по уровню. Любой из данных выводов сохраняет активность, даже если он настроен на вывод. Данная особенность может использоваться для программной генерации прерывания.

Таблица 3.3

Расширенный регистр маски прерывания ETIMSK

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	–	–	TICIE 3	OCIE 3A	OCIE 3B	TOIE 3	OCIE 3C	TOIE 3C
Чтение/запись	Чт	Чт	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Таблица 3.4

Расширенный регистр маски прерывания таймера-счетчика EICRB

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	ICS71	ICS70	ICS61	ICS60	ICS51	ICS50	ICS41	ICS40
Чтение/запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Внешние прерывания 7–4 активизируются через внешние выводы INT7–INT4, если установлены флаг I в регистре статуса SREG и соответствующая маска прерывания в регистре EIMSK. Условие, по которому генерируется прерывание, выбирается исходя из данных табл. 3.5. Для определения фронтов на выводах INT7–INT4 осуществляется выборка их состояний. Если выбрано прерывание по фронту или изменению уровня, то прерывание будет сгенерировано при условии, что на входе появляется импульс, длительность которого больше одного периода синхронизации. При действии на входе более коротких импульсов генерация прерывания не гарантируется. Необходимо обратить внимание, что частота синхронизации ЦПУ может быть ниже, чем частота XTAL, если разрешена работа делителя частоты XTAL. Если выбрано прерывание по низкому уровню, то прерывание генерируется при условии, что до момента окончания выполнения текущей инструкции на входе по прежнему присутствует низкий уровень. Если разрешено прерывание по уровню, то оно будет генерироваться непрерывно до тех пор, пока на входе присутствует низкий уровень.

Таблица 3.5

Задание условия генерации запроса на прерывание

ISCN1	**ISCN0**	**Описание**
0	0	Низкий уровень на INTn генерирует запрос на прерывание
0	1	Любое изменение логического состояния на INTn генерирует запрос на прерывание
1	0	Падающий фронт, выявленный по двум выборкам на INTn, генерирует запрос на прерывание
1	1	Нарастающий фронт, выявленный по двум выборкам на INTn, генерирует запрос на прерывание

Листинг программы

```
.include "m128def.inc"
.dseg
        .org    $0100
.cseg
        .org    $0000
        jmp     Init
        .org    $000e                   ;Вектор прерываний по входу PE6 (INT6)
        jmp     int_666
Init:
        ldi r16,low(ramend)
        out spl,r16
        ldi r16,high(ramend)
        out sph,r16
; Установка условия генерации прерывания по заднему фронту
        ldi r16,0b00100000
        out EICRB,r16
        ldi r16,0b01000000              ;Регистр маски внешних прерываний
        out EIMSK,r16
        clr r16
        cbi ddre,6
        sei
        gogogo:
        nop
        nop
        nop
        nop
        nop
        jmp gogogo
        int_666:
        ldi r16,1
        ldi r17,1
        ldi r18,1
        reti
```

При пошаговом выполнении для генерации прерывания необходимо установить и сбросить 6-й бит PINE (рис. 3.1).

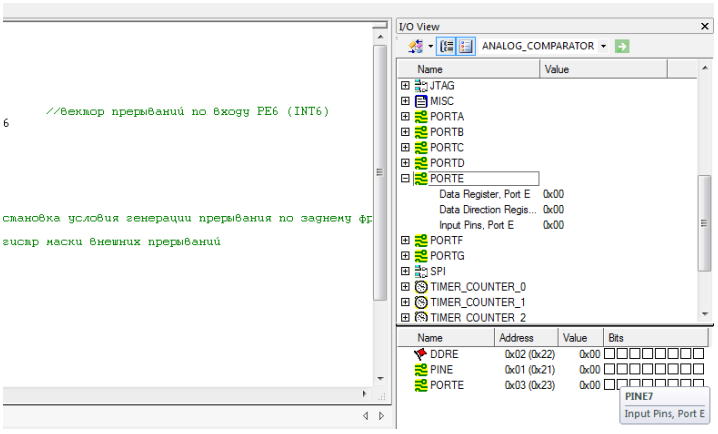


Рис. 3.1. Окно I/O View

Практическая часть

3.5. Контрольные вопросы

1. Назовите примеры применения подпрограмм.
2. Опишите синтаксис написания подпрограммы и укажите, какие команды в ней используются?
3. Назначение стека при работе с подпрограммой и порядок его инициализации.
4. Назовите отличия команды CALL от RCALL.
5. Как определяется приоритет при обработке прерываний и как он влияет на работу микроконтроллера?
6. Перечислите основные векторы сброса и прерываний.
7. Какие действия производит микроконтроллер непосредственно перед обработкой прерывания?
8. Назовите примеры применения прерываний в программировании микроконтроллера.

3.6. Варианты заданий

1. Загрузить в регистр R16 число 73521. Вычислить результат от деления содержимого регистра на 3. Для оптимизации пользоваться подпрограммами.
2. Загрузить в регистр R18 число 48411. Вычислить результат от деления содержимого регистра на 3. Для оптимизации пользоваться подпрограммами.
3. Загрузить в регистр R20 число 76758. Вычислить результат от деления содержимого регистра на 6. Для оптимизации пользоваться подпрограммами.
4. Загрузить в регистр R22 число 4716. Вычислить результат от деления содержимого регистра на 6. Для оптимизации пользоваться подпрограммами.
5. Загрузить в регистр R17 число 95123. Вычислить результат от деления содержимого регистра на 7. Для оптимизации пользоваться подпрограммами.

6. При появлении положительного фронта на порте ввода-вывода PE4 найти произведение чисел 24 и 41, при отрицательном фронте на входе – вычесть из результата 100.

7. При появлении положительного фронта на порте ввода-вывода PE4 найти произведение чисел 27 и 38, при отрицательном фронте на входе – вычесть из результата 56.

8. При высоком уровне сигнала на входе PE6 инкрементировать R17, при низком – уменьшить значение на 2.

9. При высоком уровне сигнала на входе PD1 инкрементировать R18, при низком – уменьшить значение на 2.

10. При переполнении таймера-счетчика 0 записать 100 в регистр R16.

11. При переполнении таймера-счетчика 0 записать 255 в регистр R21.

12. При появлении положительного фронта на входе PE5 запустить таймер-счетчик 0.

13. При появлении положительного фронта на входе PD3 запустить таймер-счетчик 0.

14. При появлении отрицательного фронта на входе PE7 увеличивать значение в стеке на 1.

15.При появлении отрицательного фронта на входе PD0 увеличивать значение в стеке на 1.

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №4
ИССЛЕДОВАНИЕ УСТРОЙСТВА ДИНАМИЧЕСКОЙ ИНДИКАЦИИ

Цель работы. Ознакомиться с документацией на устройство динамической индикации, изучить схему сопряжения микроконтроллера с устройством динамической индикации, разработать и отладить программу вывода информации на устройство динамической индикации.

Перед началом выполнения лабораторной работы необходимо изучить следующие вопросы:

– способы построения цифровой индикации;
– устройства динамической индикации;
– устройства статической индикации.

В ходе выполнения работы необходимо:

– изучить электрическую принципиальную схему к лабораторной работе;
– разработать программу в соответствии с индивидуальным заданием;
– отладить программу в среде Atmel «AVRStudio»;
– загрузить программу в учебный стенд НТЦ-31.100;
– исследовать работу динамической индикации в соответствии с индивидуальным заданием;
– оформить отчет.

После выполнения лабораторной работы необходимо ответить на контрольные вопросы.

Теоретическая часть

4.1. Устройства цифровой индикации

Для отображения цифровой индикации наиболее распространены светодиодные 7-сегментнтые индикаторы (рис. 4.1).

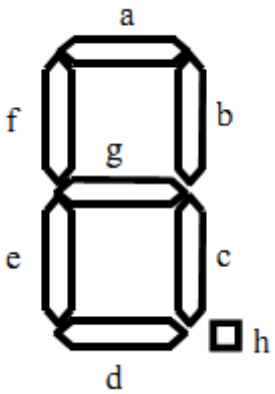
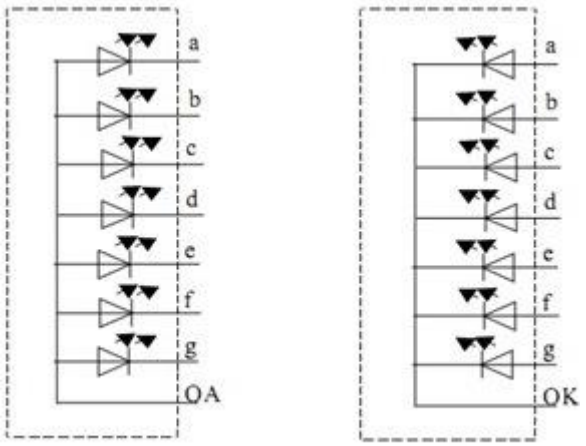


Рис. 4.1. 7-сегментный символьный индикатор

Сегменты индикатора расположены в виде восьмерки. Иногда добавляют еще один восьмой сегмент – десятичную запятую. Засвечивая группы сегментов, можно получить все цифры и некоторые символы. Конструктивно индика-

торы оформляются в виде светодиодных модулей с общим катодом или с общим анодом, как в учебном стенде НТЦ-31.100 (рис. 4.2).



а б

Рис. 4.2. Электрические принципиальные схемы 7-сегментных индикаторов: а – с общим анодом; б – с общим катодом

При построении систем отображения информации различают два подхода: статическая и динамическая индикация.

Статическая индикация состоит в постоянной засветке каждого индикатора от одного источника информации (рис. 4.3). В такой системе каждый индикатор подключен через свой дешифратор и регистр-защелку (R1–R3) к шине данных. Каждый регистр адресуется с помощью устройства дешифрации адреса (DC1–DC2). Такая схема предполагает значительные аппаратные затраты, так как на каждый индикатор необходим по крайней мере один регистр.

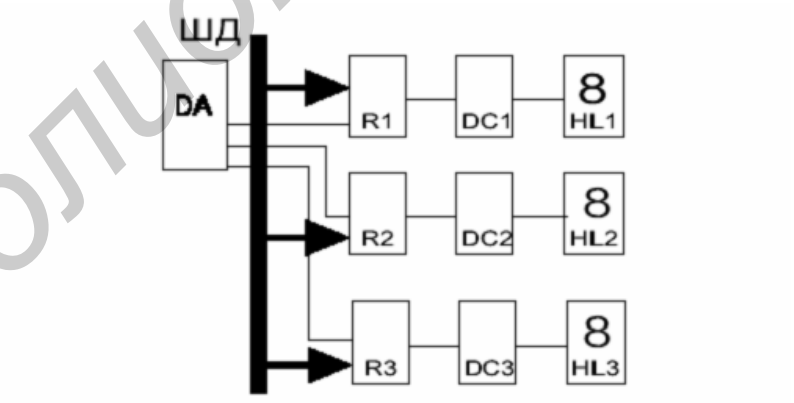


Рис. 4.3. Устройство статической индикации

Сущность динамической индикации заключается в поочередном циклическом подключении каждого индикатора к источнику данных (RD) (рис. 4.4). При использовании такой схемы включения значительно сокращаются аппаратные затраты. Но при этом необходимо обеспечить достаточное время свечения для того, чтобы не уменьшалась яркость свечения индикаторов. С другой

стороны, необходимо обеспечить достаточно быстрое переключение индикаторов, чтобы не было заметно мерцания. В обоих случаях 7-сегментные дешифраторы из схемы индикации можно исключить, а функцию дешифрации переложить на микроконтроллер, что, с одной стороны, несколько усложнит программную реализацию индикации, но при этом можно выводить на индикацию не только цифры, но и другие символы.

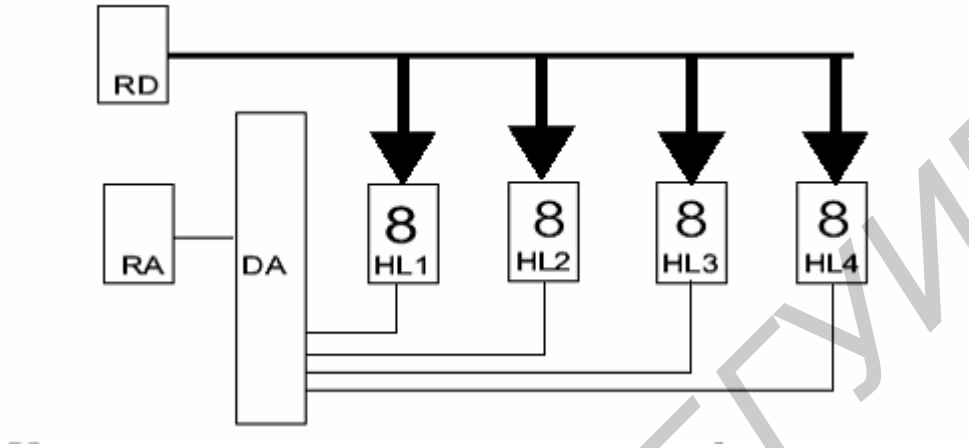


Рис. 4.4. Устройство динамической индикации

4.2. Электрическая принципиальная схема

В прил. 2 изображена электрическая принципиальная схема к данной лабораторной работе.

В схеме использован параллельный регистр DD2 1594ИР37 (74ACT574) для подключения к микроконтроллеру сегментов символьного светодиодного индикатора. Для организации параллельной шины данных используется 8-битный порт PA микроконтроллера. Для защелкивания информации в регистре используется порт PC.3. Для переключения разрядов индикаторов ССИ используются транзисторные ключи VT1–VT8, управляемые выходами микросхемы DD3 дешифратора HC138. Для управления дешифратором используются выходы порта микроконтроллера PC.0 – PC.2.

В стенде НТЦ-31.100 организована последовательная шина. Для выбора конкретного устройства на шине используется набор сигналов CS0–CS7. Эти сигналы формируются на выходах дешифратора DD4 74HC138 (К1564ИД7). Для формирования кода выбираемого устройства используются порты PB.0, PB.4 и PB.5. Для защелкивания информации из сдвигового регистра в выходной регистр-защелку в 74HC595 применяется сигнал LC (где активный фронт – передний), а для формирования этого сигнала – сигнал CS2. Для ввода информации с дискретных источников (тумблеров SW1–SW8) используется микросхема DD13 сдвигового регистра с параллельной загрузкой HCT165 (К5564ИР9). Порт PB.3 применяется для приема информации в последовательной форме из микроконтроллера, а порт PB.1 – для тактирования сдвигового регистра.

4.3. Пример программы «бегущая» строка

```
.include "m128def.inc"
.equ   Sgm_A =    0b01111111
.equ   Sgm_B =    0b10111111
.equ   Sgm_D =    0b11011111
.equ   Sgm_E =    0b11101111
.equ   Sgm_F =    0b11110111
.equ   Sgm_C =    0b11111011
.equ   Sgm_H =    0b11111101
.equ   Sgm_G =    0b11111110
.cseg
        .org            0
        jmp             INIT

.equ   SymbolB =  (Sgm_A&Sgm_C&Sgm_D&Sgm_E&Sgm_F&Sgm_G)
.equ   SymbolS =  (Sgm_A&Sgm_E&Sgm_F)
.equ   SymbolU =  (Sgm_B&Sgm_C&Sgm_D&Sgm_F&Sgm_G)
.equ   SymbolI =   (Sgm_B&Sgm_C&Sgm_D&Sgm_E&Sgm_F)
.equ   SymbolR =  (Sgm_A&Sgm_B&Sgm_E&Sgm_F&Sgm_G)

String:         .DB             ;Отображаемый на дисплее текст
0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff,SymbolR,SymbolI,SymbolU,SymbolS,Symbol
B,0xFF,0xFF,0xff,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xff,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,0xFF,0xff,0xFF,0xFF,0xFF,0xFF,0xFF

 INIT:
        ldi     R16,0xFF
        out     ddrA,R16      ;Настройка порта A на передачу
        ldi     R16,0b00001111
        out     ddrC,R16    ;Настройка младшей тетрады порта C на передачу
;Предварительный сброс вспомогательных регистров и регистров-счетчиков
        clr     R17
        clr     R18
        clr     r19
        clr     r20
        clr     r25
        clr      r24
        clr     r23
        clr     r22
        clr     r21
        ldi     R28,low(String)     ;Инициализируется массив
        ldi     R29,high(String)
        lsl     R28
```

```
        rol     R29
        ldi r20,0x1f        ;Перемещаем указатель массива в его конец
        add r28,r20
Start:
        out     portC,R17   ;Задаем номер сегмента
        mov     R30,R28     ;Копируем адрес элемента массива в регистр Z
        mov     R31,R29
        add     R30,R17
        adc     R31,R18

        lpm     R2,Z        ;Копируем элемент массива с адресом Z в регистр R2
        out     portA,R2    ;Задаем элемент массива в сегмент
        cbi     portC,3     ;Задаем отрицательный фронт на синхровходе
                            ;дешифратора для зажигания диодов
        sbi     portC,3
        dec     R19         ;Создаем задержку
        brne    PC-1

        ldi     R16,0xFF    ;Гасим сегмент
        out     portA,R16
        cbi     portC,3
        sbi     portC,3
        inc     R17         ;Увеличиваем номер сегмента
        andi    R17,7       ;Ограничиваем номер (не более семи)

        dec     r25         ;Создаем задержку движения строки
        brne    pc-1
        dec     r24
        brne    pc-1
        dec     r23
        brne    pc-1
        dec     r22
        brne    pc-1
        dec     r21
        brne    start

        dec     r28         ;Уменьшаем адрес на единицу
        dec     r20         ;Уменьшаем счетчик адреса на единицу
        brne    start       ;Пока счетчик адреса не обнулится – возвращаться на
                            ;старт
        ldi     r20,0x1f    ;Задаем верхнее значение счетчика
        add     r28,r20     ;Возвращаем указатель на конец массива
        jmp     start       ;Возвращаемся на старт
```

Вид исполняемой программы в симуляторе Proteus показан на рис. 4.5.

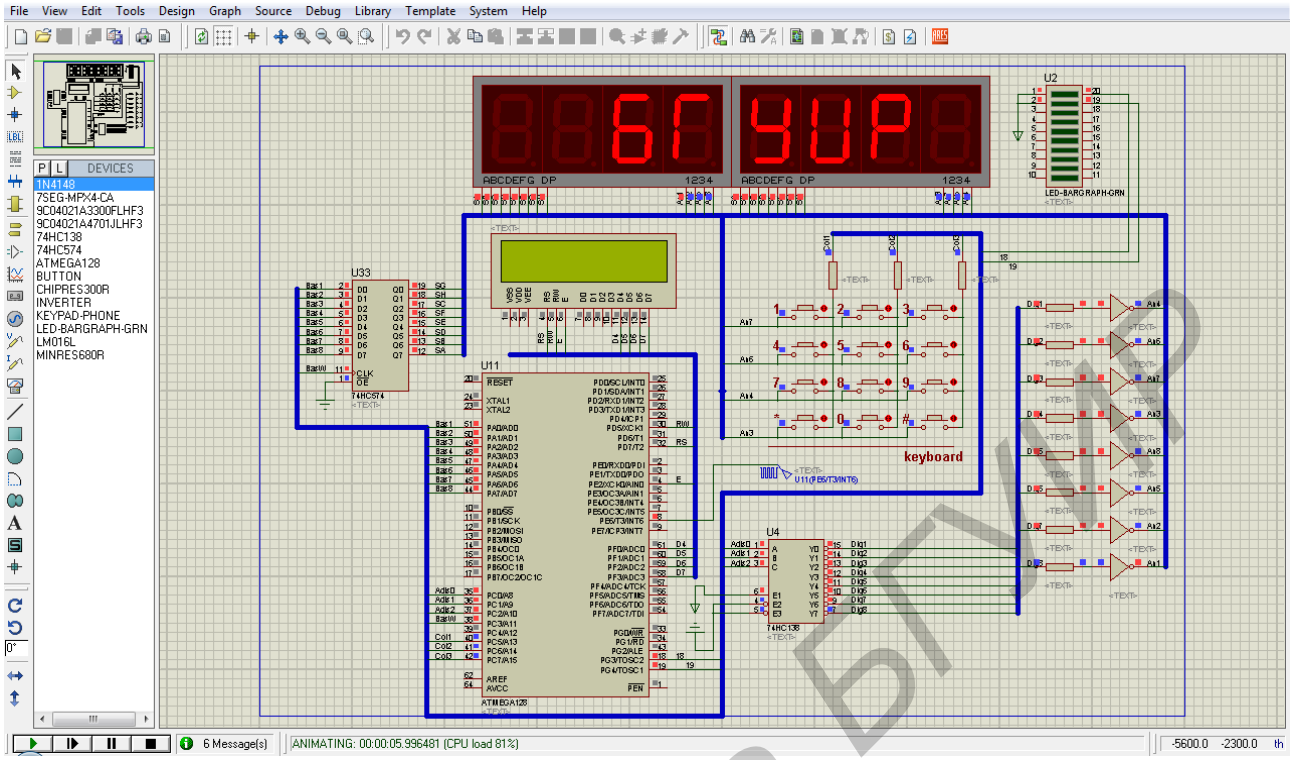


Рис. 4.5. Окно симулятора Proteus

Практическая часть

4.4. Контрольные вопросы

1. Каков принцип построения светодиодных индикаторов?
2. Поясните сущность статической индикации.
3. Поясните сущность динамической индикации.
4. Каков порядок инициализации микроконтроллера для работы с 7-сегментным дисплеем в стенде НТЦ 31.100?
5. Приведите пример последовательности вывода символов на 7-сегментный дисплей.
6. Каково назначение дешифратора в схеме вывода информации на 7-сегментный дисплей?
7. Как реализуется «бегущая» строка?

4.5. Варианты заданий

Разработать программу для учебного стенда НТЦ-31.100, выводящую на светодиодный 7-сегментный индикатор «бегущую» строку, содержащую следующую информацию:

1. Последовательность цифр от 0h до Fh с пробегом строки справа налево.

2. Последовательность цифр от Fh до 0h с пробегом строки справа налево.

3. Последовательность цифр от 0h до Fh с пробегом строки слева направо.

4. Последовательность цифр от Fh до 0h с пробегом строки слева направо.

5. «BSUIR-xxxxxx», где xxxxxx – номер вашей группы с пробегом строки справа налево.

6. «BSUIR-xxxxxx», где xxxxxx – номер вашей группы с пробегом строки слева направо.

7. «BSUIR-xxxxxx», где xxxxxx – текущая дата в формате ДД.ММ.ГГГГ с пробегом строки слева-направо.

8. «BSUIR-xxxxxx», где xxxxxx – текущая дата в формате ГГГГ.ММ.ДД с пробегом строки справа налево.

9. Число π с точностью до десятого знака с пробегом строки справа налево.

10. Число *e* с точностью до десятого знака с пробегом строки справа налево.

11. Квадратный корень числа 2 с точностью до 5-го знака с пробегом строки справа налево.

12. Последовательность чисел, разделяемых пробелом и являющихся степенями числа 2 от первой до пятой с пробегом строки справа налево.

13. Последовательность чисел, разделяемых пробелом и являющихся степенями числа 2 от первой до пятой с пробегом строки слева направо.

14. Значение счетчика числа пробегов строки с обнулением на 10 и пробегом слева направо.

15. Значение счетчика числа пробегов строки с обнулением на 10 и пробегом слева направо.

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №5
ИССЛЕДОВАНИЕ УСТРОЙСТВА МАТРИЧНОЙ ЖИДКОКРИСТАЛЛИЧЕСКОЙ ИНДИКАЦИИ

Цель работы. Ознакомиться с документацией устройства матричной жидкокристаллической индикации (ЖКИ), изучить схему сопряжения микроконтроллера с устройством матричной индикации, разработать и отладить программу вывода информации в устройство матричной ЖКИ.

Теоретические сведения

5.1. Теоретические основы и принципы работы ЖКИ

Жидкокристаллические индикаторы управляют отражением и пропусканием света для создания изображений цифр, букв, символов и т. д. В отличие от светодиодов жидкокристаллические индикаторы не излучают свет. Основу ЖКИ составляют жидкие кристаллы (ЖК), молекулы которых упорядочены послойно определенным образом между двумя стеклянными пластинами. В каждом слое сигарообразные молекулы ЖК выстраиваются в одном направлении, их оси становятся параллельными (рис. 5.1).

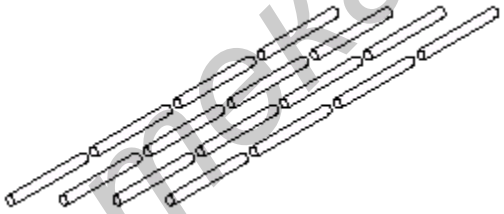


Рис. 5.1. Один слой молекул ЖК

Стеклянные пластины имеют специальное покрытие, такое, что направленность молекул в двух крайних слоях перпендикулярна. Ориентация каждого слоя ЖК плавно изменяется от верхнего к нижнему, формируя спираль (рис. 5.2). Эта спираль «скручивает» поляризацию света по мере его прохождения через дисплей.

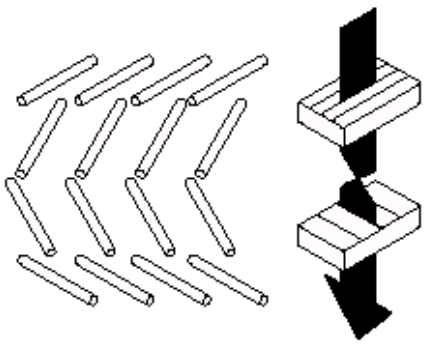


Рис. 5.2. Несколько слоев молекул ЖК

Под действием электрического поля молекулы ЖК переориентируются параллельно полю. Этот процесс называется твистнематическим полевым эффектом. При такой ориентации поляризация света не скручивается при прохождении через слой ЖК (рис. 5.3). Если передний поляризатор ориентирован перпендикулярно заднему, свет пройдет через включенный дисплей, но заблокируется задним поляризатором. В этом случае ЖКИ действует как заслонка свету. Отображение различных символов достигается избирательным травлением проводящей поверхности, предварительно созданной на стекле. Не вытравленные области становятся символами, а вытравленные – фоном дисплея.

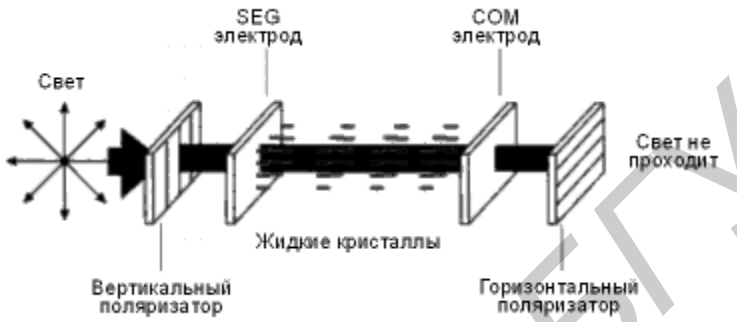


Рис. 5.3. «Включенное» состояние ЖКИ

Символы создаются из одного или нескольких сегментов. Каждый сегмент может быть адресован (запитан) индивидуально, чтобы создать отдельное электрическое поле. Таким образом, прохождение света управляется электрически, включая и отключая необходимые сегменты. В неактивной части дисплея направленность молекул остается спиральной, формируя фон. Запитанные сегменты составляют символы, контрастирующие с фоном. В зависимости от ориентации поляризатора, ЖКИ может отображать позитивное или негативное изображение. В дисплее с позитивным изображением передний и задний поляризаторы перпендикулярны друг другу так, что незапитанные сегменты и фон пропускают свет с измененной поляризацией, а запитанные препятствуют прохождению света. Результат – темные символы на светлом фоне. В дисплее с негативным изображением поляризаторы параллельны, т. е. находятся «в фазе» друг с другом, и препятствуют прохождению света с повернутой поляризацией так, что незапитанные символы и фон темные, а запитанные – светлые.

Рефлективный ЖКИ имеет отражатель (рефлектор) за задним поляризатором, который отражает свет, прошедший через незапитанные сегменты и фон. В негативных рефлективных дисплеях свет отражается через запитанные, «включенные» сегменты. Трансмиссивные дисплеи используют те же принципы, но фон или сегменты становятся ярче за счет использования задней подсветки.

5.2. Рефлективные индикаторы

Обычно рефлективные ЖКИ (работающие на отражение) используют режим отображения с темными символами на светлом фоне (так называемое позитивное изображение). В индикаторе с позитивным изображением передний и задний поляризаторы находятся в противофазе, или перекрестно поляризованы на 90°. Если сегмент «выключен», внешний свет идет по следующему пути: проходит через вертикальный поляризатор, через прозрачный электрод сегмента, через ЖК молекулы, которые скручивают его на 90°, через прозрачный общий электрод, через горизонтальный поляризатор, а затем попадает на рефлектор, который посылает свет обратно по тому же пути (рис. 5.4).

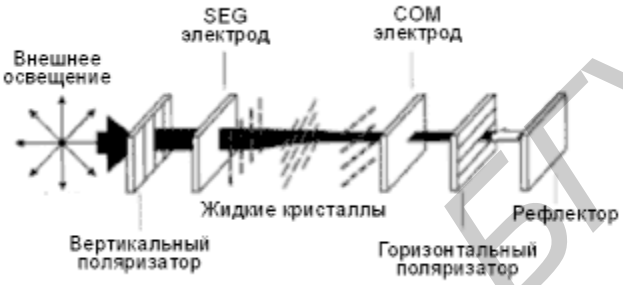


Рис. 5.4. Рефлективный индикатор в выключенном состоянии

Если сегмент «включен», внешний свет не изменяет своей поляризации при проходе через слой жидких кристаллов. Таким образом, поляризация света противоположна заднему поляризатору, что не дает свету пройти к отражателю. Так как свет не отражается, получается темный сегмент (рис. 5.5).

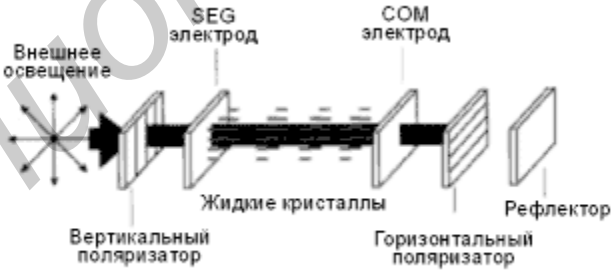


Рис. 5.5. Рефлективный индикатор во включенном состоянии

Рефлективные индикаторы очень яркие, с отличным контрастом и имеют широкий угол обзора. Они требуют хорошего внешнего освещения и не используют искусственную заднюю подсветку (хотя в некоторых моделях применяют подсветку сверху). Благодаря малым токам потребления рефлективные индикаторы часто используются в устройствах с питанием от батареек.

5.3. Трансмиссивные индикаторы

Трансмиссивные ЖКИ (работающие на пропускание) не отражают свет. Напротив, они создают изображение, управляя светом искусственного источника освещения, расположенного позади индикатора. В трансмиссивных индикаторах передний и задний поляризаторы находятся «в фазе» друг с другом (параллельны). В выключенном сегменте поляризованный свет подсветки скручивается на 90° молекулами ЖК и оказывается в противофазе с передним поляризатором. Поляризатор блокирует свет, создавая темный сегмент.

Если сегмент включен, свет не скручивается, оказываясь «в фазе» с передним поляризатором, и проходит через него, создавая световой рисунок. Таким образом трансмиссивный дисплей создает светлое изображение на темном фоне (негативное изображение).

Трансмиссивные индикаторы должны иметь заднюю подсветку, чтобы гарантировать равномерное свечение сегментов. Они хороши для использования в условиях приглушенного или слабого освещения. В условиях прямого солнечного света подсветка не может преодолеть солнечных лучей и изображение не заметно.

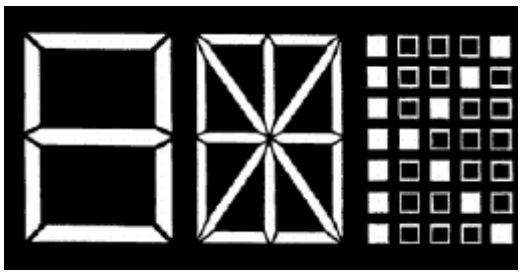
5.4. Трансрефлективные индикаторы

Трансрефлективные (работающие на пропускание и отражение) индикаторы используют белый или серебряный полупрозрачный материал, который отражает часть внешнего света, а также пропускает свет задней подсветки. Поскольку эти индикаторы как отражают, так и пропускают свет, они могут использоваться в широком диапазоне яркостей освещения.

Примером могут служить индикаторы мобильных телефонов. Они читаемы как при ярком свете, так и в полной темноте. Трансфлективные дисплеи имеют более низкую контрастность по сравнению с рефлективными, так как часть света проходит сквозь отражатель.

5.5. Сегменты ЖКИ

Части ЖКИ, работающие как заслонки, включаясь и выключаясь для формирования изображений, называются сегментами. Сегменты создаются прозрачными электродами из оксидов индия и олова, нанесенными на стекло ЖКИ. Цифры от 0 до 9 и некоторые буквы могут быть отображены на 7-сегментном индикаторе. 16-сегментный индикатор может отобразить цифры, все латинские и почти все русские буквы (кроме Й, Ц, Щ). Для того чтобы символы были менее угловатыми и более натуральными, используют матричные индикаторы. С их помощью можно также отображать небольшие изображения. Количество сегментов индикатора (рис. 5.6) влияет на метод управления им.



а б в

Рис. 5.6. Дисплей:
а – 7-сегментный; б – 16-сегментный; в – матричный 5×7

ЖКИ обычно имеет время срабатывания 50 мс при 20 °С, а лучшие модели – до 10 мс. Стандартный ЖКИ может отображать сигнал до 10 Гц, если это требуется. Невооруженным глазом тяжело отследить данные с такой частотой.

5.6. Жидкокристаллический индикатор SC-1602 BULT

Микросхема SC-1602 BULT представляет собой жидкокристаллический матричный индикатор (ЖКИ) 2 строки по 16 символов со встроенным контроллером. Условное обозначение микросхемы приведено на рис. 5.7, а назначение выводов – в табл. 5.1.

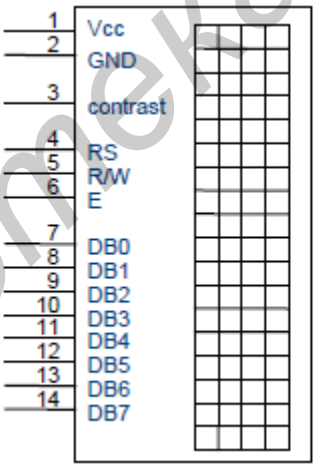


Рис. 5.7. Условное обозначение ЖКИ SC-1602 BULT

Таблица 5.1

Назначение выводов микросхемы SC-1602 BULT

Обозначение выводов	Номер вывода	Назначение
VCC	1	Питание
GND	2	Земля
Contrast	3	Регулировка контраста
RS	4	Выбор регистра ("1" – регистр данных, "0" – регистр команд)

Окончание табл. 5.1

Обозначение выводов	Номер вывода	Назначение
R/W	5	Выбор регистра ("1" – чтение данных, "0" – запись данных)
E	6	Сигнал разрешения чтения/записи
DB0–DB7	7…14	0 – 7 биты данных

Основные параметры ЖКИ SC-1602 BULT:
– размер символов 5×7 точек;
– встроенный контроллер HD44780 или совместимый с ним;
– напряжение питания 5 В.

Структура индикатора приведена на рис. 5.8, таблица символов для шрифта кириллицы – на рис. 5.9.

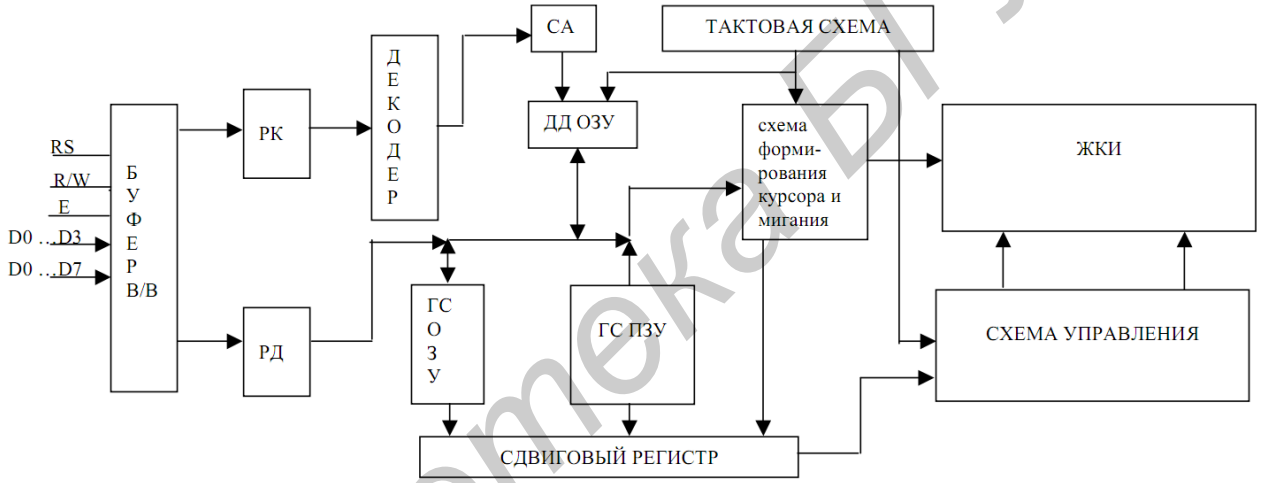


Рис. 5.8. Структура ЖКИ SC-1602 BULT

ЖКИ состоит из следующих структурных элементов:
– регистр данных;
– регистр команд (РК);
– ПЗУ генератора символов (ГС) (ПЗУ содержит таблицу символов. Каждый символ имеет 8-битный код. Всего в ПЗУ содержится 192 символа);
– ОЗУ генератора символов (ОЗУ содержит символы пользователя);
– счетчик адреса;
– ОЗУ данных дисплея (ДД) (ОЗУ содержит информацию для вывода на дисплей);
– схема формирования курсора и мигания.

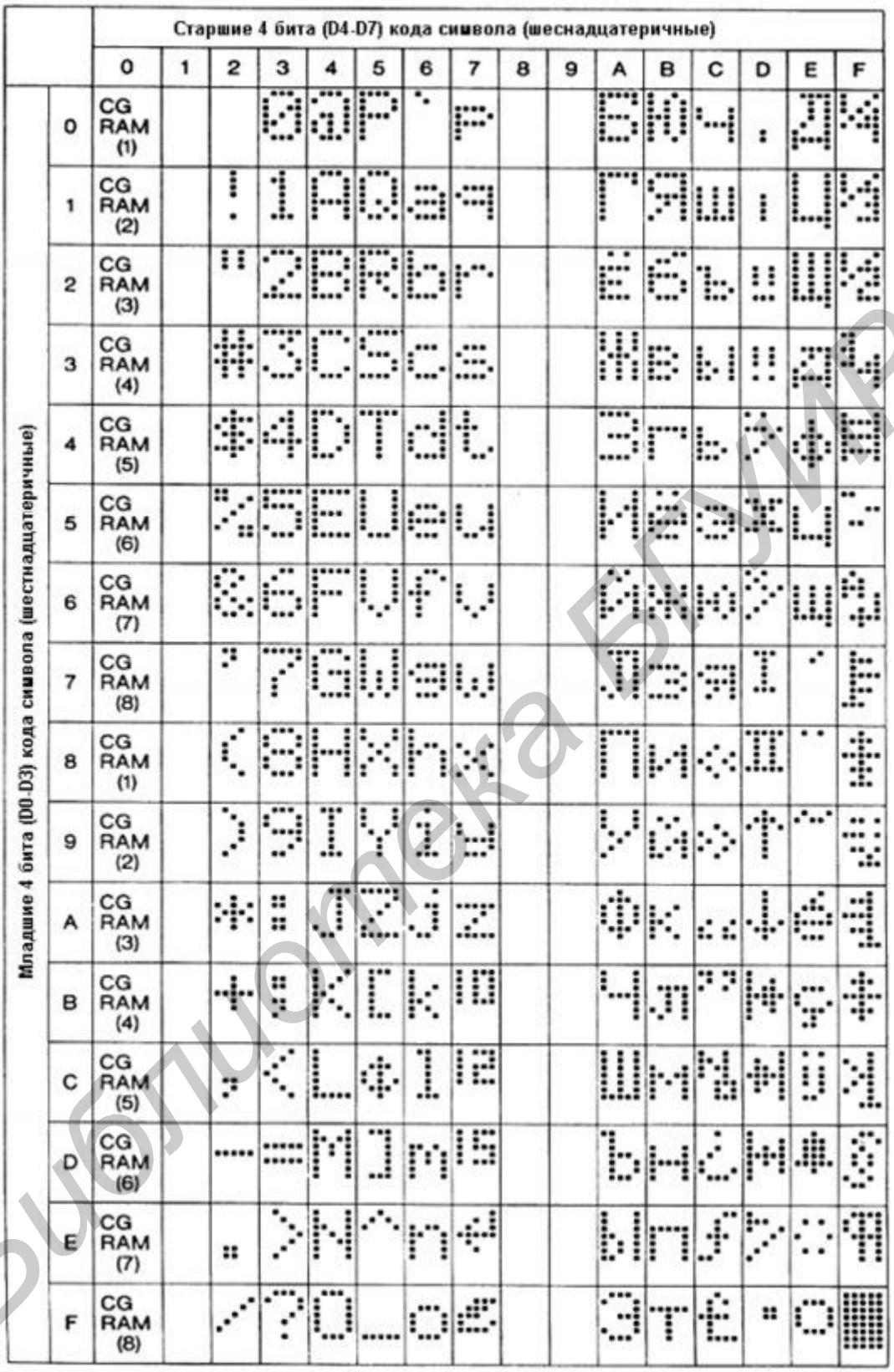


Рис. 5.9. Таблица символов для шрифта кириллицы

Буфер ввода-вывода может работать в 4-битном и в 8-битном режиме. В 4-битном режиме данные в индикатор передаются тетрадами. Перед началом работы с индикатором он должен быть проинициализирован. Последовательность инициализации при работе в 4-битном режиме приведена на рис. 5.10. Таблица команд работы с индикатором приведена на рис. 5.11.

Рис. 5.10. Последовательность инициализации

Команда	Код										Описание	Время выполнения
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Очистка индикатора	0	0	0	0	0	0	0	0	0	1	Очистка дисплея и перевод курсора в начальную позицию	82 μs - 1.64ms
Возврат в начало	0	0	0	0	0	0	0	0	1	*	Перевод курсора или смещенного дисплея в начальную позицию	40 μs - 1.6ms
Установка режима ввода	0	0	0	0	0	0	0	1	I/D	S	Установка направления перемещения курсора и разрешение дисплея	40 μs
Вкл/выкл индикатора	0	0	0	0	0	0	1	D	C	B	Вкл/выкл дисплея (D) или курсора (C) и установка режима мерцания символа	40 μs
Сдвиг курсора и индикатора	0	0	0	0	0	1	S/C	R/L	*	*	Перемещение курсора и сдвиг дисплея без изменения DD RAM	40 μs
Установка режимов	0	0	0	0	1	DL	N	F	*	*	Установка 8/4-битного режима, количества строк и шрифта	40 μs
Установка адреса CG RAM	0	0	0	1	ACG						Установка адреса CG RAM	40 μs
Установка адреса DD RAM	0	0	1	ADD							Установка адреса DD RAM	40 μs
Чтение флага BUSY и адреса	0	1	BF	AC							Чтение флага BUSY и содержимого счетчика адреса	1 μs
Запись данных в CG или DD RAM	1	0	Write Data								Запись данных в DD RAM или CG RAM	43 μs
Чтение данных из CG или DD RAM	1	1	Read Data								Чтение данных из DD RAM или CG RAM	43 μs
	I/D=1:Увеличение I/D=0:Уменьшение S=1:Сопровождает сдвиг дисплея S/C=1:Сдвиг дисплея S/C=0:Смещение курсора R/L=1:Сдвиг вправо R/L=0:Сдвиг влево DL=1:8 бит DL=0:4 бита N=1:2 строки N=0:1 строка F=1:5x10 точек F=0:5x7 точек BF=1:Занят BF=0:Доступ разрешен										DD RAM:RAM данных дисплея CG RAM:RAM генератора символов ACG:CG RAM адрес ADD:DD RAM адрес Соответствует адресу курсора AC:Счетчик адреса	

Рис. 5.11. Таблица команд

5.7. Электрическая принципиальная схема

В электрической принципиальной схеме стенда НТЦ-31.100 (рис. 5.12) организована последовательная шина. Для выбора конкретного устройства на шине используется набор сигналов CS0–CS7. Эти сигналы формируются на выходах дешифратора DD4 74HC138 (К1564ИД7). Для формирования кода выбираемого устройства используются порты PB.0, PB.4 и PB.5.

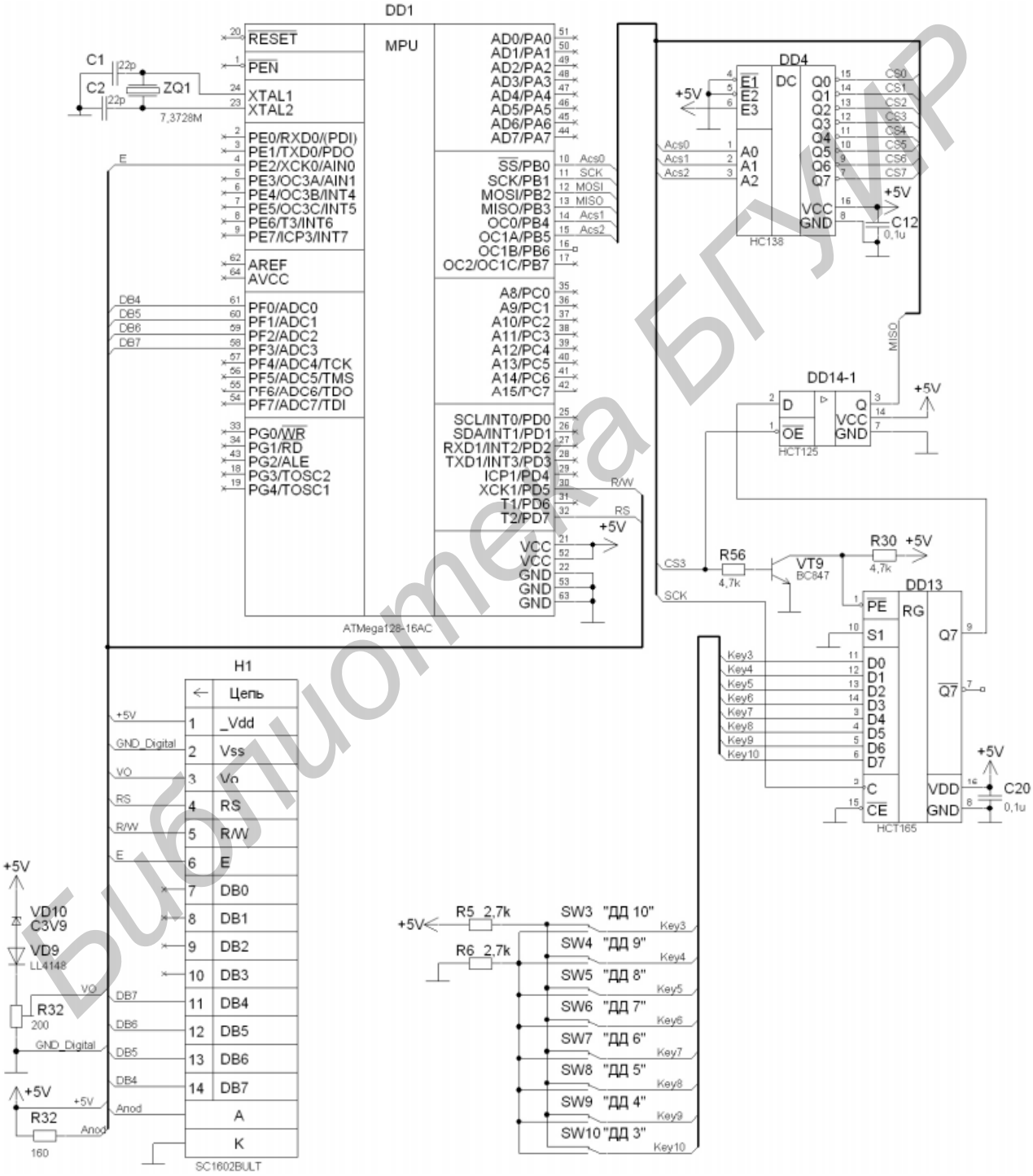


Рис. 5.12. Электрическая принципиальная схема к лабораторной работе

Для ввода информации с дискретных источников (тумблеров SW1–SW8) используется микросхема DD13 сдвигового регистра с параллельной загрузкой HCT165 (К5564ИР9). Порт PB.3 служит для приема информации в последовательной форме из микроконтроллера, порт PB.1 – для тактирования сдвигового регистра.

Для управления жидкокристаллическим индикатором используются линии PF.0–PF.3, PE.2, PD.5 и PD.7 микроконтроллера.

Практическая часть

5.8. Контрольные вопросы

1. Каков физический принцип работы ЖК индикаторов?
2. Поясните принцип работы рефлективных ЖК индикаторов.
3. Поясните принцип работы трансмиссивных ЖК индикаторов.
4. Поясните принцип работы трансрефлективных ЖК индикаторов.
5. Какое условное графическое обозначение и назначение выводов ЖК дисплея SC-1602 BULT применяется в настоящее время?
6. Опишите последовательность инициализации ЖК дисплея SC-1602 BULT при работе в 4-битном режиме.
7. Из чего состоит схема подключения ЖК дисплея к портам ввода-вывода микроконтроллера?
8. Каков алгоритм вывода символов на ЖК дисплей?

5.9. Варианты заданий

Разработать программу для учебного стенда НТЦ-31.100, выводящую на матричный жидкокристаллический индикатор (МЖКИ) свою фамилию и номер варианта на первой строке и следующую информацию на второй:

1. Результат операции логического «И» между регистром R20, равным 15h, и R21, равным AFh.
2. Результат операции логического «ИЛИ» между регистром R20, равным EBh, и R21, равным 3Fh.
3. Результат операции логического «ИЛИ-НЕ» между регистром R20, равным D0h, и R21, равным AAh.
4. Результат операции логического «И-НЕ» между регистром R20, равным 56h, и R21, равным 6Dh.
5. Побитовую инверсию числа AFh, представленную в двоичном виде.
6. Инверсию числа 231, представленную в двоичном виде.
7. Число 01111011b, представленное в 16-ричном виде.
8. Число 11110001b, представленное в 16-ричном виде.
9. Бегущую строку, содержащую информацию о текущей дате и номере вашей группы.

10. Бегущую строку, содержащую информацию о текущем дне недели и номере вашей группы.

11. Бегущую строку, содержащую информацию о текущем месяце и номере вашей группы.

12. Бегущую строку, содержащую символы №20, №4F, №23 и №FF, с последовательным полным заполнением строки каждым из них.

13. Бегущую строку, содержащую символы №20, №2D, №2B и №2A, с последовательным полным заполнением строки каждым из них.

14. Бегущую строку, содержащую расшифровку аббревиатуры названия изучаемой дисциплины.

15. Бегущую строку, содержащую счетчик числа циклов пробега строк.

5.10. Пример выполнения лабораторной работы

Задача

Разработать программу для учебного стенда НТЦ-31.100, позволяющую отобразить на МЖКИ строки «БГУИР – ПИКС» и «2014» в центре верхней и нижней строки соответственно.

Решение

Программа для решения этой задачи будет состоять из блока инициализации работы МЖКИ и блока вывода информации. Процедуру вывода символа на дисплей, а также установки курсора в начало первой и второй строки вынесем в отдельные подпрограммы **lcd_print_symbol**, **lcd_1st_line** и **lcd_2nd_line** соответственно.

Текст программы

```
include     "m128def.inc"
.cseg
        .org    0000
        jmp     Init
Init:
        ldi     R16,0x00        ;Формируем стек, начиная с ячейки памяти 0x400
        out     SPL,R16         ;Младший байт
        ldi     R17,0x04
        out     SPH,R17         ;Старший байт
;Инициализируем дисплей
        sbi     DDRD,5          ;Устанавливаем на вывод линию R/W
        sbi     DDRD,7          ;Устанавливаем на вывод линию RS
        sbi     DDRE,2          ;Устанавливаем на вывод линию E
        ldi     R16,0b00001111  ;Устанавливаем на вывод линии DB7…DB4
        sts     DDRF,R16
        cbi     PortD,5         ; R/W = 0 режим записи в ЖК дисплей (Write)
;Задержка
        clr     r13
```

```
        clr     r14
        dec     R13
        brne    PC-1
        dec     R14
        brne    PC-3
        cbi     PortD,7         ;RS = 0 – установка дисплея на прием команд
        ldi     R16,0b0011      ;Загружаем DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 1
        sts     PortF,R16       ;Отправляем это на ЖК дисплей
        sbi     PortE,2         ;Включаем режим приема
        clr     r13             ;Очищаем используемый регистр
        dec     r13             ;Уменьшаем на единицу
        brne    pc-1
        cbi     PortE,2         ;Выключаем режим приема
;Задержка
        dec     R13
        brne    PC-1
        dec     R14
        brne    PC-3
        ldi     R16,0b0011      ;Отправляем DB7=0,DB6=0,DB5=1.DB4=1
        sts     PortF,R16
        sbi     PortE,2         ;Включаем прием
;Задержка
        call    delay
        cbi     PortE,2         ;Выключаем прием
        call    delay
        ldi     R16,0b0011      ;Отправляем DB7 = 0,DB6 = 0,DB 5= 1, DB4=1
        sts     PortF,R16
        sbi     PortE,2
;Задержка
        call    delay
        cbi     PortE,2
        ldi     R16,0b0010      ;Установка 4-битного интерфейса:
                                ;DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 0
        sts     PortF,R16       ;Отправляем
        sbi     PortE,2         ;Включаем прием
;Задержка
        call    delay
        cbi     PortE,2         ;Выключаем прием
        ldi     R16,0b0010      ;Подтверждение 4-битного интерфейса
                                ;DB7 = 0, DB6 = 0, DB5 = 1, DB4 = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
```

```
;Установка символьной матрицы 5 x 7 пикселей
        ldi     R16,0b1000      ;N = 1, F = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Гашение дисплея
        ldi     R16,0b0000      ;DB7 = 0, DB6 = 0, DB5 = 0, DB4 =0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b1000      ;Отправляем D = 0, C = 0, B = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Установка режима ввода
        ldi     R16,0b0000
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b0110      ;Отправляем I/D = 1, S = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Включение индикатора
        ldi     R16,0b0000
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b1111      ;Отправляем D = 1, C = 1, B = 1
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Конец инициализации
start:
        call    lcd_1st_line    ;Установка курсора в левом верхнем углу
        ldi     r20,0x20
        call    lcd_print_symbol ;Запись символа " " 0x20
```

```
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0xA0        ;Запись символа "Б" 0xA0
        call    lcd_print_symbol
        ldi     r20,0xA1        ;Запись символа "Г" 0xA1
        call    lcd_print_symbol
        ldi     r20,0xA9        ;Запись символа "У" 0xA9
        call    lcd_print_symbol
        ldi     r20,0xA5        ;Запись символа "И" 0xA5
        call    lcd_print_symbol
        ldi     r20,0x50        ;Запись символа "Р" 0x50
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x2D        ;Запись символа "-" 0x  2D
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0xA8        ;Запись символа "П" 0xA8
        call    lcd_print_symbol
        ldi     r20,0xA5        ;Запись символа "И" 0xA5
        call    lcd_print_symbol
        ldi     r20,0x4B        ;Запись символа "К" 0x4B
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа "С" 0x43
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
;Установка курсора в начале второй строки
        call    lcd_2nd_line
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20        ;Запись символа " " 0x20
        call    lcd_print_symbol
```

```
        ldi     r20,0x32            ;Запись символа "2" 0x32
        call    lcd_print_symbol
        ldi     r20,0x30            ;Запись символа "0" 0x30
        call    lcd_print_symbol
        ldi     r20,0x31            ;Запись символа "1" 0x31
        call    lcd_print_symbol
        ldi     r20,0x33            ;Запись символа "3" 0x33
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
        ldi     r20,0x20            ;Запись символа " " 0x20
        call    lcd_print_symbol
;Блок задержки
        ldi     r19, 100
        dec     r17
        brne    pc-1
        dec     r18
        brne    pc-3
        dec     r19
        brne    pc-5
        jmp     start
;Блок подпрограмм работы с дисплеем:
lcd_print_symbol:           ;Запись символа на ЖК дисплей, код которого
                            ;записан в R20
        sbi     PortD,7     ;RS = 1 – установка дисплея на прием символов
        mov     r16,r20     ;Копируем выводимый символ в R16
        andi    r16,0xF0    ;Удаляем младшую тетраду из R16
        swap    r16         ;Меняем местами тетрады
        sts     PortF,R16   ;Отправляем эту тетраду на дисплей
        sbi     PortE,2     ;Включаем прием
;Задержка
        clr     r13
        dec     r13
        brne    pc-1
        cbi     PortE,2     ;Выключаем прием
        mov     r16,r20     ;Снова копируем в R16
```

```
        andi    r16,0x0F        ;Удаляем старшую тетраду
        sts     PortF,R16       ;Отправляем ее на дисплей
        sbi     PortE,2         ;Включаем прием
;Задержка
        clr     r13
        dec     r13
        brne    pc-1
        cbi     PortE,2         ;Выключаем прием
        ret                     ;Выход из подпрограммы
lcd_1st_line:
;Установка курсора в левом верхнем углу 0x80
        cbi     PortD,7         ;RS = 0 – установка дисплея на прием команд
        ldi     R16,0x8         ;Отправляем старшую тетраду команды
        sts     PortF,R16       ;Отправляем ее
        sbi     PortE,2         ;Активируем прием дисплеем
        call    delay           ;Задержка
        cbi     PortE,2         ;Закрываем прием
        ldi     R16,0x0         ;Загружаем младшую тетраду команды
        sts     PortF,R16       Оотправляем
        sbi     PortE,2         ;Активируем прием
        call    delay           ;Ожидаем
        cbi     PortE,2         ;Завершаем прием
        sbi     PortD,7         ;Установка дисплея на прием символов
        ret
lcd_2nd_line:
;Установка курсора в начале второй строки 0xC0
        cbi     PortD,7
        ldi     R16,0xC
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0x0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        sbi     PortD,7
        ret
;Конец программы
```

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №6
ИССЛЕДОВАНИЕ ВВОДА ИНФОРМАЦИИ ПРИ ПОМОЩИ КЛАВИАТУРЫ

Цель работы. Ознакомиться со схемами подключения клавиатуры к микроконтроллеру, изучить схему сопряжения микроконтроллера с клавиатурой, разработать и отладить программу обработки событий клавиатуры.

Теоретические сведения

6.1. Матричная организация клавиатур

Клавиатуры предназначаются для ввода информации в микроконтроллерное устройство, т. е. для организации интерфейса с человеком. Фактически клавиатуры представляют собой наборы дискретных переключателей. Особенностью клавиатур является достаточно большое количество таких переключателей (обычно более 10). Большее количество переключателей требует использования большего количества линий параллельных портов.

Одним из наиболее распространенных способов уменьшения требуемых линий для подключения клавиатур к параллельным портам является организация клавиатур по принципу матричного шифратора, в узлах которого размещены коммутационные элементы – клавиши (рис. 6.1).

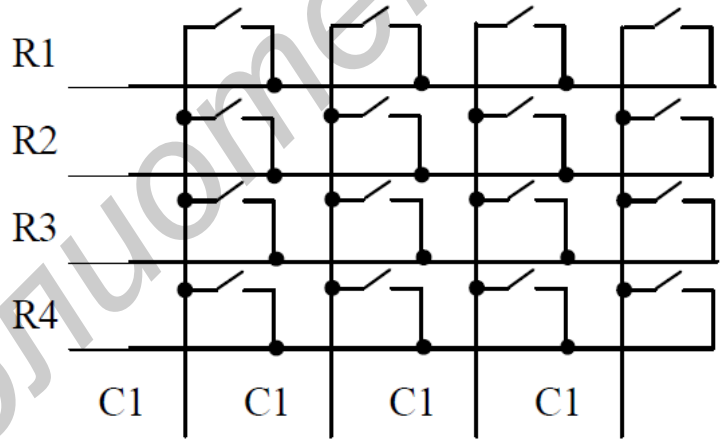


Рис. 6.1. Матричная организация клавиатур

Из показанного на рис. 6.1 примера видно, что 16-клавишную клавиатуру можно подключить, используя только 8 линий параллельного порта.

Работает матричная клавиатура следующим образом. На одну из линий R1–R4 подаются сигналы. Затем происходит сканирование одного из рядов клавиатуры. Если ни одна клавиша не нажата, то сигнал на линиях C1–C4 отсутствует. Если клавиша на сканируемом ряду нажата, то на соответствующей линии C1–C4 появляется сигнал. Таким образом, зная какой ряд в данный мо-

мент сканируется и на какой из линий C1–C4 появился сигнал, можно определить, какая клавиша нажата в данный момент.

Для организации клавиатуры используются кнопки S1–S12, включенные по матричной схеме. Сигналы выбора ряда матрицы формируются на транзисторных ключах VT1, VT3, VT5 и VT7, управляемых выходами Q0–Q3 дешифратора DD3 HC138. Для управления дешифратором используются выходы портами микроконтроллера PC.0–PC.2. Для считывания состояния нажатых клавиш используются порты PC.5–PC.7.

Для управления жидкокристаллическим индикатором используются линии PF.0–PF.3, PE.2, PD.5 и PD.7 микроконтроллера.

Практическая часть

6.2. Контрольные вопросы

1. Каков принцип организации матричной клавиатуры?
2. Приведите схему подключения клавиатуры к микроконтроллеру.
3. Опишите порядок работы микроконтроллера с клавиатурой.
4. Опишите порядок инициализации микроконтроллера для работы с клавиатурой.
5. Как формируется код нажатой клавиши?

6.3. Варианты заданий

Разработать программу для учебного стенда НТЦ-31.100, позволяющую вводить данные с клавиатуры, выполнять определенные действия над данными и выводить поясняющие надписи и результат вычислений на матричный жидкокристаллический индикатор (МЖКИ) или на дискретный светодиодный индикатор (ДСИ).

1. На ДСИ, разделяя пробелами, отобразить вводимые с клавиатуры два двузначных числа и результат их сложения в 16-ричном виде.
2. На МЖКИ, разделяя соответствующими знаками арифметических операций, отобразить вводимые с клавиатуры два двузначных числа и результат их сложения в 16-ричном виде.
3. На ДСИ, разделяя пробелами, отобразить вводимые с клавиатуры два двузначных числа и результат их вычитания (с учетом знака) в 16-ричном виде.
4. На МЖКИ, разделяя соответствующими знаками арифметических операций, отобразить вводимые с клавиатуры два двузначных числа и результат их вычитания (с учетом знака) в 16-ричном виде.
5. На МЖКИ, разделяя соответствующими знаками арифметических операций, отобразить вводимые с клавиатуры два двузначных числа и результат их произведения в 16-ричном виде.

6. На ДСИ, разделяя пробелами, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «И» между ними в 16-ричном виде.

7. На МЖКИ, разделяя соответствующими знаками логических операций, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «И» между ними в 16-ричном виде.

8. На ДСИ, разделяя пробелами, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «ИЛИ» между ними в 16-ричном виде.

9. На МЖКИ, разделяя соответствующими знаками логических операций, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «ИЛИ» между ними в 16-ричном виде.

10. На ДСИ, разделяя пробелами, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «И-НЕ» между ними в 16-ричном виде.

11. На МЖКИ, разделяя соответствующими знаками логических операций, отобразить вводимые с клавиатуры два двузначных числа и результат операции логического «И-НЕ» между ними в 16-ричном виде.

12. На ДСИ отобразить введенное число и результат его побитовой инверсии.

13. На МЖКИ отобразить введенное число и результат его побитовой инверсии.

14. На ДСИ отобразить бегущую строку, содержащую введенное с клавиатуры число.

15. На МЖКИ отобразить бегущую строку, содержащую введенное с клавиатуры число.

6.4. Пример выполнения лабораторной работы

Задача

Разработать программу для учебного стенда НТЦ-31.100, позволяющую выводить на дискретный светодиодный индикатор (ДСИ) последовательность вводимых с клавиатуры чисел.

Решение

Программа для решения этой задачи будет состоять из блока инициализации работы портов ввода-вывода микроконтроллера, блока вывода информации на ДСИ, блока сканирования клавиатуры и формирования кода нажатой клавиши.

Так как строки клавиатуры и сегмент ДСИ задаются одинаковым способом – через линии PC0–PC2, то будет производиться одновременное задание сегмента на ДСИ и сканирование соответствующей строки клавиатуры. Для хранения введенных символов используется область памяти 0x200…0x207, ад-

рес которой хранится в регистре X (для вывода символа на дисплей) и регистра Y (для сохранения символа в память).

Для того чтобы считать информацию о нажатой клавише, нужно проверить содержимое регистра PinC – на его три первых разряда будет приходить информация о состоянии трех столбцов клавиатуры. Таким образом, объединяя посылку (на portC) и принятую информацию (из pinC), получим следующие коды клавиш:

1) клавиша * – 0x23;
2) клавиша 0 – 0x43;
3) клавиша # – 0x83;
4) клавиша 1 – 0x22;
5) клавиша 2 – 0x42;
6) клавиша 3 – 0x82;
7) клавиша 4 – 0x21;
8) клавиша 5 – 0x41;
9) клавиша 6 – 0x81;
10) клавиша 7 – 0x20;
11) клавиша 8 – 0x40;
12) клавиша 9 – 0x80.

Текст программы

```
;Описание кодов сегментов на ДСИ
.include "m128def.inc"
.equ   Sgm_A =0b01111111
.equ   Sgm_B =0b10111111
.equ   Sgm_D =0b11011111
.equ   Sgm_E =0b11101111
.equ   Sgm_F =0b11110111
.equ   Sgm_C =0b11111011
.equ   Sgm_H =0b11111101
.equ   Sgm_G =0b11111110
.cseg
.org   0x300              ;Организация начала программы с ячейки памяти 300h
       jmp   INIT

;Описание кодов символов на ДСИ
.equ   Symbol0 =  (Sgm_A&Sgm_B&Sgm_C&Sgm_D&Sgm_E&Sgm_F)
.equ   Symbol1 =(Sgm_B&Sgm_C)
.equ   Symbol2 =(Sgm_a&Sgm_b&Sgm_D&Sgm_e&Sgm_G)
.equ   Symbol3 =(Sgm_A&Sgm_B&Sgm_c&Sgm_d&Sgm_G)
.equ   Symbol4 =(Sgm_B&Sgm_C&Sgm_F&Sgm_G)
.equ   Symbol5 =(Sgm_A&Sgm_C&Sgm_D&Sgm_F&Sgm_G)
.equ   Symbol6 =(Sgm_A&Sgm_C&Sgm_D&Sgm_E&Sgm_F&Sgm_G)
```

```
.equ   Symbol7 =(Sgm_A&Sgm_B&Sgm_C)
.equ   Symbol8 =(Sgm_A&Sgm_B&Sgm_C&Sgm_D&Sgm_E&Sgm_F&Sgm_G)
.equ   Symbol9 =(Sgm_A&Sgm_B&Sgm_C&Sgm_D&Sgm_F&Sgm_G)
.equ   SymbolA =(Sgm_A&Sgm_B&Sgm_C&Sgm_E&Sgm_F&Sgm_G)
.equ   SymbolB =(Sgm_C&Sgm_D&Sgm_E&Sgm_F&Sgm_G)

INIT:                   ;Блок инициализации
;Инициализация портов ввода-вывода контроллера

;Порт A инициализируется на вывод
        ldi     R16,0xFF
        out     ddrA,R16

;Младшая тетрада порта C инициализируется на ввод
        ldi     R16,0b00001111
        out     ddrC,R16

;Формирование области памяти для хранения образов нажатых клавиш
        ldi     r26,0x00    ;Регистр X – для вывода на дисплей
        ldi     r27,0x02
        ldi     r28,0x00    ;Регистр Y – для записи нажатой клавиши в память
        ldi     r29,0x02

;Вспомогательные регистры:
        clr     r20
        clr     r21
        clr     r22
        clr     r23

;Заполнение памяти образов кодом пустого символа:
        ldi     r16,0xff    ;Запись образа пустого символа
        st      x+,r16      ;Отправка образа в память
        andi    r26,7       ;Повторяем 8 раз
        brne    pc-3
start:                      ;Вывод информации на дисплей
        out     portC,R26   ;Отправка номера разряда
        ld      r25,x       ;Загрузка образа выводимого символа
        out     portA,R25   ;Отправка образа на порт
        cbi     portC,3     ;Формирование отрицательного фронта
        sbi     portC,3
        dec     r20         ;Задержка
        brne    pc-1
        ldi     R25,0xFF    ;Гашение индикатора
        out     portA,R25
```

```
        cbi     portC,3
        sbi     portc,3
        in      R16,pinC    ;Загрузка из порта C информации о нажатии клавиши
                            ;Маски для предотвращения установки лишних бит
        andi    r16,0b11100000
        andi    r26,0b00001111
        or      r16,r26     ;Соединение регистра отправки и регистра приема
        mov     r17,r16     ;Копирование этого кода во вспомогательный регистр

;Проверка наличия сигнала нажатой клавиши
        andi    r17,0b11100000
        brne    signal      ;Если получен не ноль – сигнал есть, если ноль – нет
        inc     r26         ;Переход на следующий разряд дисплея и столбца
                            ;клавиатуры
        andi    r26,7       ;Ограничение – не больше 8
        jmp     start       ;Возврат на старт
signal:                     ;Если сигнал с клавиатуры есть:
        cpi     r16,0x43    ;Сравниваем полученный код с кодом клавиши 0
        breq    zero        ;Если код совпал, переход на запись образа, если не
                            ;совпал – проверяем дальше
        cpi     r16,0x22
        breq    one
        cpi     r16,0x42
        breq    two
        cpi     r16,0x82
        breq    three
        cpi     r16,0x21
        breq    four
        cpi     r16,0x41
        breq    five
        cpi     r16,0x81
        breq    six
        cpi     r16,0x20
        breq    seven
        cpi     r16,0x40
        breq    eight
        ldi     r16,0x80
        breq    nine
        ldi     r16,0x23    ;*
        breq    ten
        ldi     r16,0x83    ;#
        breq    eleven      ;Если ни один код не совпал – возвращение на старт
        inc     r26         ;Переход на следующий разряд дисплея\столбца
                            ;клавиатуры
```

```
        andi    r26,7           ;Ограничение – не больше 8
jmp start
ten:                            ;Запись кода нажатой клавиши в память
        ldi     r25,symbolA
        jmp     delay           ;Переход на блок сдвига и задержки
eleven:
        ldi     r25,symbolB
        jmp     delay
zero:
        ldi     r25,symbol0
        jmp     delay
one:
        ldi     r25,symbol1
        jmp     delay
two:
        ldi     r25,symbol2
        jmp     delay
three:
        ldi     r25,symbol3
        jmp     delay
four:
        ldi     r25,symbol4
        jmp     delay
five:
        ldi     r25,symbol5
        jmp     delay
six:
        ldi     r25,symbol6
        jmp     delay
seven:
        ldi     r25,symbol7
        jmp     delay
eight:
        ldi     r25,symbol8
        jmp     delay
nine:
        ldi     r25,symbol9
        jmp     delay
delay:                          ;Блок сдвига и задержки
        ldi     r28,0x07        ;Установка указателя в конец массива чисел
        ldi     r21,0x08        ;Установка счетчика
        ld      r20,y+          ;Загрузка символа из массива
        st      y,r20           ;Запись символа на следующее место
        dec     r28             ;Уменьшение указателя на 2 позиции
```

```
        dec     r28
        dec     r21             ;Уменьшение счетчика
        brne    pc-5            ;Повторяем 8 раз. Сдвиг массива на 1 позицию вправо
        ldi     r28,0x00        ;Устанавливаем указатель на нулевую ячейку
        st      y,r25           ;Записываем образ нажатой клавиши в нулевую ячейку
        ldi     r22,8
        dec     r20
        brne    pc-1
        dec     r21
        brne    pc-3
        dec     r22
        brne    pc-5
        jmp     start
;Конец программы
```

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №7
ИССЛЕДОВАНИЕ ИНТЕРФЕЙСА ПРИБОРНОЙ ШИНЫ TWI И РАБОТЫ ЧАСОВ РЕАЛЬНОГО ВРЕМЕНИ

Цель работы. Ознакомиться с интерфейсом приборной шины TWI, изучить схему сопряжения микросхемы часов реального времени с микроконтроллером, разработать и отладить программу для обслуживания устройства.

Теоретическая часть

7.1. Двухпроводной последовательный интерфейс TWI

Двухпроводной последовательный интерфейс TWI (I2C) идеально подходит для типичных применений микроконтроллера. Протокол TWI позволяет проектировщику системы внешне связать до 128 различных устройств через одну двухпроводную двунаправленную шину, где одна линия – линия синхронизации SCL и одна – линия данных SDA. В качестве внешних аппаратных компонентов, которые требуются для реализации шины, необходим только подтягивающий к плюсу питания резистор на каждой линии шины.

Все устройства, которые подключены к шине, имеют свой индивидуальный адрес, а механизм определения содержимого шины поддерживается протоколом TWI (рис. 7.1).

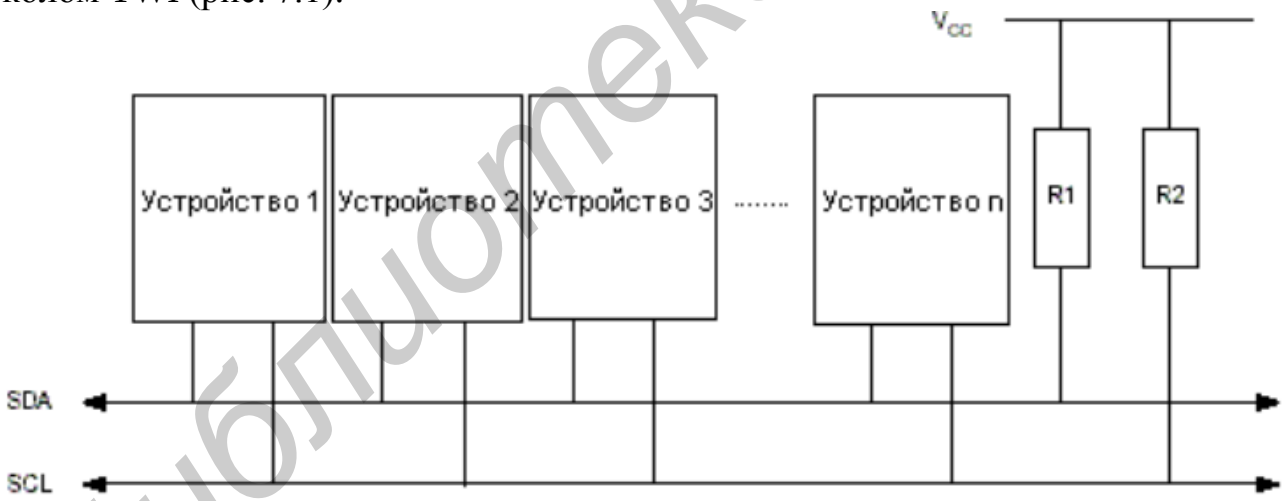


Рис. 7.1. Внешние подключения к шине TWI

Как показано на рис. 7.1, обе линии шины подключены к положительной шине питания через подтягивающие резисторы. Среди всех совместимых с TWI устройствами в качестве драйверов шины используются транзистор или с открытым стоком или с открытым коллектором. Этим реализована функция монтажного «И», которая очень важна для двунаправленной работы интерфейса. Низкий логический уровень на линии шины TWI генерируется, если одно или более из TWI-устройств выводит логический «0». Высокий уровень на линии присутствует, если все TWI-устройства перешли в третье высокоимпедансное

состояние, позволяя подтягивающим резисторам задать уровень логической «1». Необходимо обратить внимание, что при подключении к шине TWI нескольких AVR-микроконтроллеров, для работы шины должны быть запитаны все из этих микроконтроллеров.

Количество устройств, которое может быть подключено к одной шине ограничивается предельно допустимой емкостью шины (400 пФ) и 7-разрядным адресным пространством.

7.2. Формат посылки и передаваемых данных

7.2.1. Передаваемые биты

Каждый передаваемый бит данных по шине TWI сопровождается импульсом на линии синхронизации. Уровень данных должен быть стабильным, когда на линии синхронизации присутствует логическая «1».

Исключением для этого правила является генерация условий старта и останова сеанса связи.

7.2.2. Условия СТАРТа и ОСТАНОВа

Ведущее устройство инициирует и заканчивает передачу данных. Передача инициируется, когда ведущий формирует условие СТАРТа на шине, и прекращается, когда ведущий формирует на шине условие ОСТАНОВа. Между условиями СТАРТа и ОСТАНОВа шина считается занятой и в этом случае никакой другой мастер не может осуществлять управляющие воздействия на шине. Существуют особые случаи, когда новое условие СТАРТа возникает между условиями СТАРТа и ОСТАНОВа. Данный случай именуется как условие «Повторного старта» и используется при необходимости инициировать мастером новый сеанс связи, не теряя при этом управление шиной. После «Повторного старта» шина считается занятой до следующего ОСТАНОВа. Это идентично поведению после СТАРТа, следовательно, при описании ссылка на условие СТАРТа распространяется и на «Повторный старт», если, конечно же, нет специального примечания. Как показано на рис. 7.2, условия СТАРТа и ОСТАНОВа являются изменением логического уровня на линии SDA, когда на линии SCL присутствует логическая «1».

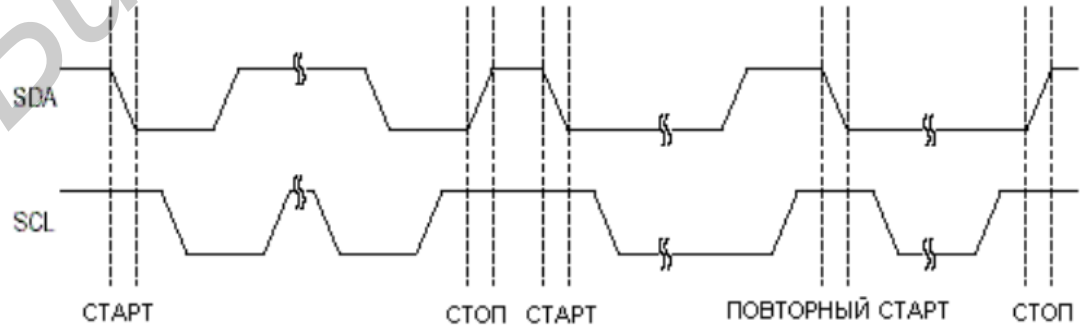


Рис. 7.2. Условия СТАРТа, ПОВТОРНОГО СТАРТа и ОСТАНОВа

7.2.3. Формат адресного пакета

Все передаваемые адресные пакеты по шине TWI состоят из 9 бит, в том числе 7 бит адреса, 1 бит управления для задания типа операции ЧТЕНИЕ/ЗАПИСЬ и 1 бит подтверждения. Если бит ЧТЕНИЕ/ЗАПИСЬ = 1, то будет выполнена операция чтения, иначе – запись. Если подчиненный распознает, что к нему происходит адресация, то он должен сформировать низкий уровень на линии SDA на 9-м цикле SCL (формирование бита подтверждения). Если адресуемое подчиненное устройство занято или по каким-либо другим причинам не может обслужить ведущее устройство, то на линии SDA необходимо оставить высокий уровень во время цикла подтверждения. Ведущий после этого может передать условие ОСТАНОВа или «Повторного старта» для инициации новой передачи. Адресный пакет, состоящий из адреса подчиненного устройства и бита ЧТЕНИЕ или ЗАПИСЬ, обозначим следующим образом: ПОДЧИН_АДР+ЧТЕНИЕ или ПОДЧИН_АДР+ЗАПИСЬ соответственно.

Старший разряд адресного байта передается первым. Нет никаких ограничений на выбор адреса подчиненного устройства, за исключением адреса 0000000, который зарезервирован для общего вызова.

При определении общего вызова все подчиненные устройства должны ответить низким уровнем на линии SDA во время цикла подтверждения (ACK). Общий вызов необходимо использовать, если одно и то же сообщение необходимо передать от ведущего к нескольким подчиненным устройствам. Если вслед за битом ЗАПИСИ передан адрес общего вызова, то все подчиненные устройства устанавливают низкий уровень на линии SDA для подтверждения общего вызова во время цикла подтверждения (рис. 7.3). Следующие пакеты данных будут приниматься всеми подчиненными устройствами, которые подтвердили общий вызов. Необходимо обратить внимание, что передача адреса общего вызова вслед за битом ЧТЕНИЕ бессмысленна, т. к. одновременное чтение нескольких подчиненных устройств одним ведущим невозможно.

Все адреса с форматом 1111xx необходимо зарезервировать для будущего использования.

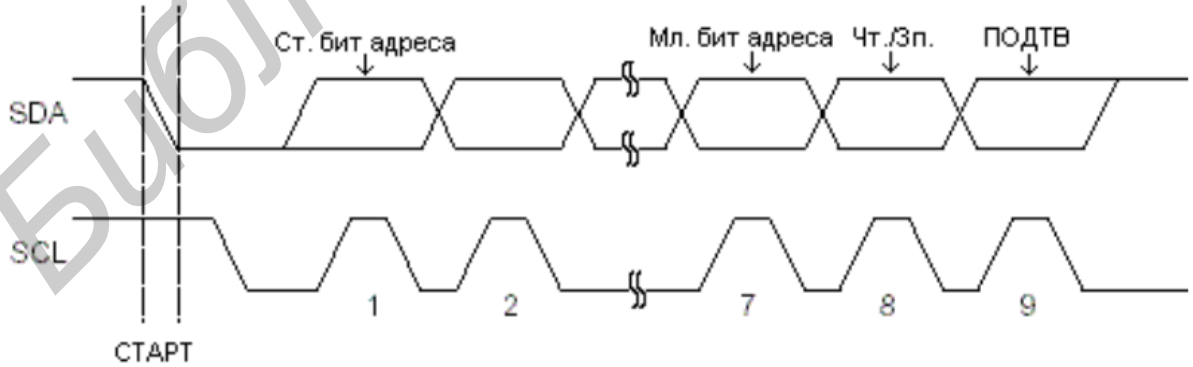


Рис. 7.3. Формат адресного пакета

7.2.4. Формат пакета данных

Все пакеты данных, передаваемые по шине TWI, состоят из 9 бит, в том числе 1 байт данных и бит подтверждения. Во время передачи данных ведущее

устройство генерирует синхронизацию, а также условия СТАРТа и ОСТАНОВа, при этом на приемник возлагается подтверждение приема.

Подтверждение (ПОДТВ) сигнализируется приемником выводом низкого уровня на линию SDA во время 9-го такта сигнала SCL. Если приемник оставляет линию SDA в высоком состоянии, то этот сигнализирует о том, что подтверждения не было (НЕТ ПОДТВ). После получения приемником последнего байта или если по каким-либо причинам не имеется возможности далее принимать данные он должен информировать передатчика отправкой бита НЕТ ПОДТВ (нет подтверждения) после последнего байта (рис. 7.4). Старший бит данных передается первым.

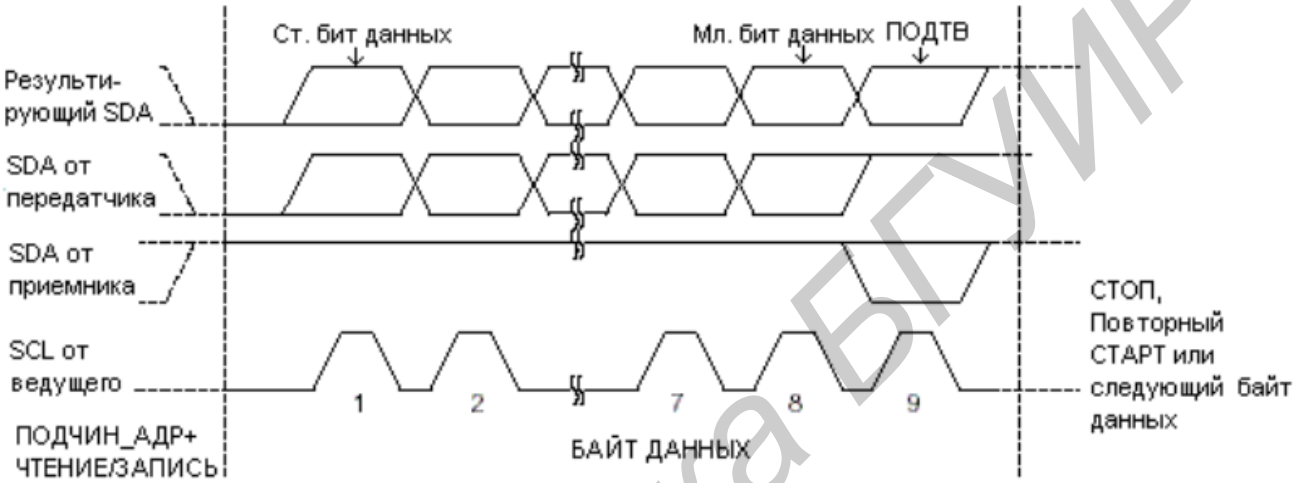


Рис. 7.4. Формат пакета данных

7.3. Модуль TWI

Модуль TWI состоит из нескольких подмодулей (рис. 7.5). Все регистры, выделенные жирной линией, доступны через шину данных микроконтроллера.

7.3.1. Выводы SCL и SDA

Выводы SCL и SDA связывают двухпроводной интерфейс микроконтроллера с остальными микроконтроллерами в системе. Драйверы выходов содержат ограничитель скорости изменения фронтов для выполнения требований к TWI. Входные каскады содержат блок подавления помех, задача которого состоит в игнорировании импульсов длительностью менее 50 нс. Необходимо обратить внимание, что к каждой из этих линий можно подключить внутренний подтягивающий резистор путем установки разрядов PORTD.0 (SCL), PORTD.1 (SDA). Использование встроенных подтягивающих резисторов в ряде случаев позволяет отказаться от применения внешних.

7.3.2. Блок генератора скорости связи

Блок генератора скорости связи управляет периодом импульсов SCL в режиме ведущего устройства. Период SCL задается регистром скорости TWI (TWBR) и значением бит управления предделителем в регистре состояния TWI

(TWSR). В подчиненном режиме значения скорости или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ подчиненного устройства должна быть минимум в 16 раз выше частоты SCL. Необходимо обратить внимание, что подчиненные могут продлевать длительность низкого уровня на линии SCL, тем самым уменьшая среднюю частоту синхронизации шины TWI. Частота SCL генерируется в соответствии со следующим выражением:

$$f_{SCL} = \frac{f_{ЦПУ}}{16 + 2 \cdot (TWBR) \cdot 4^{TWPS}},$$

где TWBR – значение регистра скорости TWI;

TWPS – значение бита предделителя в регистре состояния TWI.

TWBR должен быть равен не менее 10, если TWI работает в ведущем режиме. Если TWBR меньше 10, то ведущий может генерировать некорректное состояние на линиях SDA и SCL. Проблема возникает при работе в ведущем режиме при передаче условий СТАРТ+ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ подчиненному.

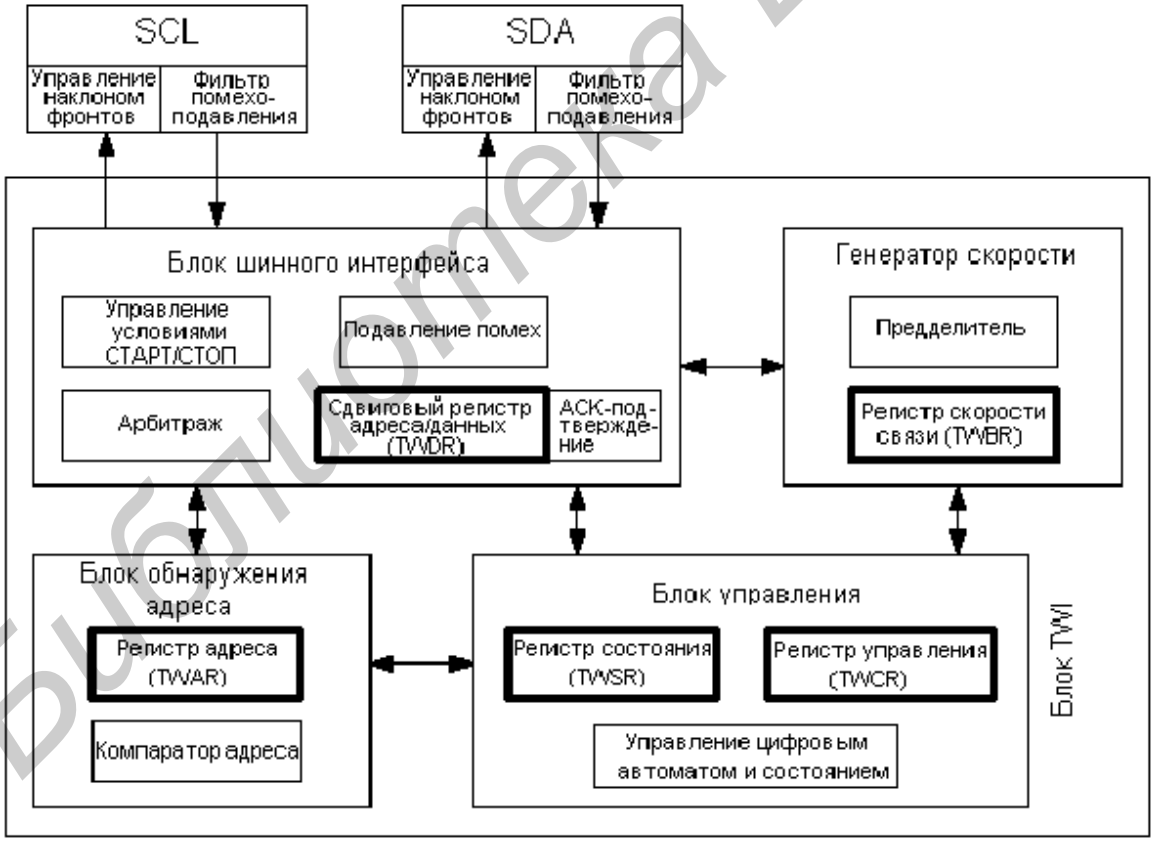


Рис. 7.5. Функциональная схема модуля TWI

7.3.3. Блок шинного интерфейса

Данный блок содержит сдвиговый регистр адреса и данных (TWDR), контроллер СТАРТа/СТОПа и схему арбитрации. TWDR содержит передаваемый

байт адреса или данных, или принятый байт адреса или данных. Помимо 8-разрядного регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита (НЕТ) ПОДТВ. К данному регистру нет прямого доступа со стороны программного обеспечения. Однако во время приема он может устанавливаться или сбрасываться путем манипуляций с регистром управления TWI (TWCR). В режиме передатчика значение принятого бита (НЕТ) ПОДТВ можно определить по значению регистра TWSR.

Контроллер СТАРТа/СТОПа отвечает за генерацию и детекцию условий СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП. Контроллер СТАРТа/СТОПа позволяет обнаружить условия СТАРТ и СТОП, даже если микроконтроллер находится в одном из режимов сна. Этим обеспечивается возможность пробуждения микроконтроллера по запросу ведущего шины.

Если TWI инициировал передачу в качестве ведущего, то схема арбитрации непрерывно контролирует передачу, определяя возможность дальнейшей передачи. Если TWI теряет арбитрацию, то блок формирует соответствующий сигнал блоку управления, который выполняет адекватные действия и генерирует соответствующий код состояния.

7.3.4. Блок обнаружения адреса

Блок обнаружения адреса проверяет, равен ли принятый адрес значению 7-разрядного адреса из регистра TWAR. Если установлен бит разрешения обнаружения общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При адресном совпадении подается сигнал блоку управления, что позволяет выполнить ему необходимые действия. В зависимости от установки регистра TWCR подтверждение адреса TWI может происходит, а может и нет. Блок обнаружения адреса способен функционировать, даже когда микроконтроллер переведен в режим сна, тем самым позволяя возобновить нормальную работу микроконтроллера по запросу мастера шины. Если при адресном совпадении TWI в экономичном режиме микроконтроллера, т. е. когда инициируется возобновление работы микроконтроллера, возникает другое прерывание (например INT0), то TWI прекращает работу и возвращается к состоянию холостого хода (Idle). Если возникновение данного эффекта нежелательно, то необходимо следить, чтобы во время обнаружения адресования, когда микроконтроллер находится в режиме выключения (Power-down), было разрешено только одно прерывание.

7.3.5. Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики в соответствии с установками регистра управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWINT. Следующим тактом обновляется содержимое регистра статуса TWI – TWSR, в котором будет записан код, идентифицирующий возникшее событие. Даная информация хранится в TWSR только

тогда, когда установлен флаг прерывания TWI. Остальное время в регистре TWSR содержится специальный код состояния, который информирует о том, что нет информации о состоянии TWI. До тех пор пока установлен флаг TWINT, линия SCL остается в низком состоянии. Этим обеспечивается возможность завершить программе все задачи перед продолжением сеанса связи.

Флаг TWINT устанавливается в следующих ситуациях:

– после передачи условия СТАРТ/ПОВТОРНЫЙ_СТАРТ;

– после передачи ПОДЧИН_АДР+ЧТЕНИЕ/ЗАПИСЬ;

– после передачи адресного байта;

– после потери арбитрирования;

– после того как TWI адресован собственным подчиненным адресом или общим вызовом;

– после приема байта данных;

– после приема условия СТОП или ПОВТОРНЫЙ_СТАРТ в режиме подчиненной адресации;

– после возникновения ошибки по причине некорректного условия СТАРТ или СТОП.

7.4. Регистры TWI

7.4.1. Регистр скорости связи шины TWI – TWBR

Регистр TWBR (табл. 7.1) задает коэффициент деления частоты генератора скорости связи. Генератор частоты скорости связи – это делитель частоты, который формирует сигнал синхронизации SCL в режимах «Ведущий».

Таблица 7.1

Разряды регистра TWBR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	TWBR 7	TWBR 6	TWBR 5	TWBR 4	TWBR 3	TWBR 2	TWBR 1	TWBR 0
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Примечание. Разряд 7…0 – биты регистра скорости связи шины TWI.

7.4.2. Регистр управления шиной TWI – TWCR

Регистр TWCR (табл. 7.2) предназначен для управления работой TWI. Он используется для разрешения работы TWI, для инициации сеанса связи ведущего путем генерации условия СТАРТ на шине, для генерации подтверждения приема, для генерации условия СТОП и для останова шины во время записи в регистр TWDR. Он также сигнализирует о попытке ошибочной записи в регистр TWDR, когда доступ к нему был запрещен.

Разряд 7 – TWINT: Флаг прерывания TWI

Данный бит устанавливается аппаратно, если TWI завершает текущее задание и ожидает реакции программы. Если бит I в SREG и бит TWIE в TWCR установлены, то микроконтроллер переходит на вектор прерывания TWI. Линия SCL остается в низком состоянии, пока установлен флаг TWINT. Флаг TWINT сбрасывается программно путем записи в него логической «1». Обратите внимание, что данный флаг сбрасывается не автоматически при переходе на вектор прерывания. Также нужно учесть, что очистка данного флага приводит к возобновлению работы TWI. Из этого следует, что программный сброс данного флага необходимо выполнить после завершения опроса регистров TWAR, TWSR и TWDR.

Таблица 7.2

Разряды регистра TWCR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт	Чт/Зп	Чт	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Разряд 6 – TWEA: Бит разрешения подтверждения

Бит TWEA управляет генерацией импульса подтверждения. Если в бит TWEA записана логическая «1», то импульс ПОДТВ генерируется на шине TWI, если выполняется одно из следующих условий:

1. Принят собственный подчиненный адрес.
2. Принят общий вызов, когда установлен бит TWGCE в регистре TWAR.
3. Принят байт данных в режиме ведущего приемника или подчиненного приемника. Запись логического «0» в бит TWEA позволяет временно отключиться от двухпроводной последовательной шины. Для возобновления распознавания адреса необходимо записать в данный бит логическую «1».

Разряд 5 – TWSTA: Бит условия СТАРТ

Программист должен установить данный бит (при необходимости стать ведущим) на двухпроводной последовательной шине. TWI аппаратно проверяет доступность шины и генерирует условие СТАРТ, если шина свободна. Однако если шина занята, то TWI ожидает появления условия СТОП, а затем генерирует новое условие СТАРТ для перехвата состояния ведущего шины. TWSTA необходимо сбрасывать программно после передачи условия СТАРТ.

Разряд 4 – TWSTO: Бит условия СТОП

Установка бита TWSTO в режиме ведущего приводит к генерации условия СТОП на двухпроводной последовательной шине. Если на шине выполняется условие СТОП, то бит TWSTO сбрасывается автоматически. В подчинен-

ном режиме установка бита TWSTO может использоваться для выхода из условия ошибки. В этом случае условие СТОП не генерируется, но интерфейс TWI возвращается к хорошо сконфигурированному безадресному подчиненному режиму и переводит линии SCL и SDA в высокоимпедансное состояние.

Разряд 3 – TWWC: Флаг ошибочной записи

Бит TWWC устанавливается при попытке записи в регистр данных TWDR, когда TWINT имеет низкий уровень. Флаг сбрасывается при записи регистра TWDR, когда TWINT = 1.

Разряд 2 – TWEN: Бит разрешения работы TWI

Бит TWEN разрешает работу TWI и активизирует интерфейс TWI. Если бит TWEN установлен, то TWI берет на себя функции управления линиями ввода-вывода SCL и SDA. При этом разрешается работа ограничителей скорости изменения фронтов и помехоподавляющих фильтров. Если данный бит равен нулю, то TWI отключается и все передачи прекращаются независимо от состояния работы.

Разряд 1 – Резервный бит

Данный бит является резервным и считывается как логический «0».

Разряд 0 – TWIE: Разрешение прерывания TWI

Если в данный бит записана логическая «1» и установлен бит I в регистре SREG, то запрос на прерывание TWI будет генерироваться до тех пор, пока установлен флаг TWINT.

7.4.3. Регистр состояния TWI – TWSR

Регистр TWSR представлен в табл. 7.3.

Таблица 7.3

Разряды регистра TWSR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт	Чт/Зп	Чт/Зп
Исходное значение	1	1	1	1	1	0	0	0

Разряды 7…3 – TWS: Состояние TWI

Данные 5 бит отражают состояние логики блока TWI и двухпроводной последовательной шины. Необходимо обратить внимание, что считываемое значение из регистра TWSR содержит и 5-разрядный код состояния и 2-разрядное значение, управляющее предделителем. Программист должен маскировать к логическому «0» биты предделителя во время проверки бита состояния. В этом случае проверка состояния не будет зависеть от настройки предделителя.

Разряд 2 – Резервный бит

Данный бит является резервным и считывается как логический «0».

Разряды 1…0 – TWPS: Биты предделителя TWI

Данные биты отличаются полным доступом (чтение/запись) и позволяют управлять предделителем скорости связи TWI (табл. 7.4).

Таблица 7.4

Предделитель скорости связи TWI

TWPS1	**TWPS0**	**Значение предделителя**
0	0	1
0	1	4
1	0	16
1	1	64

7.4.4. Регистр данных шины TWI – TWDR

В режиме передатчика регистр TWDR (табл. 7.5) содержит следующий байт для передачи. В режиме приемника регистр TWDR содержит последний принятый байт. Запись в регистр возможна, только когда TWI не выполняет процесс сдвига данных. Такое состояние наступает, когда происходит аппаратная установка флага прерывания TWINT. Необходимо обратить внимание, что регистр данных не может инициализироваться пользователем перед возникновением первого прерывания. Данные в регистре TWDR остаются стабильными, пока установлен бит TWINT. Во время сдвига последовательной передачи данных одновременно происходит сдвиг для последовательного ввода. TWDR всегда содержит последний байт, представленный на шине, исключая ситуацию возобновления нормальной работы микроконтроллера по прерыванию TWI. В этом случае состояние TWDR является неопределенным. В случае потери арбитрирования шины данные, передаваемые от ведущего к подчиненному, не теряются. Управление битом ПОДТВ происходит автоматически под управлением схемы TWI, а ЦПУ непосредственного доступа к биту ПОДТВ не имеет.

Таблица 7.5

Разряды регистра TWDR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	TWD7	TWD6	TWD5	TWD4	TWD3	TWD2	TWD1	TWD0
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

7.5. Часы реального времени

Интегральные часы реального времени – это специализированная микросхема, в состав которой входят:

− стабилизированный кварцевый генератор;

− счетчики, вырабатывающие количество секунд, часов, дней, месяцев, лет;

− управляющая логика;

− схемы внешних интерфейсов;

− схемы резервного (аккумуляторного) питания и др.

Для генератора чаще всего используется частота 32,768 кГц. Из нее легко получить при помощи двоичного делителя частоту в 1 Гц. Логика счетчиков учитывает високосные годы в широком диапазоне лет.

Аккумуляторное питание используется для обеспечения работы часов независимо от наличия питания всего устройства. Часто микросхемы оснащают дополнительными функциональными блоками: памятью, будильником и т. п.

7.5.1. Микросхема часов реального времени DS1307

Отличительные особенности микросхемы:

− подсчет реального времени в секундах, минутах, часах, датах месяца, месяцах, днях недели и годах с учетом високосности текущего года вплоть до 2100 г.;

− 56 байт энергонезависимого ОЗУ для хранения данных;

− последовательный интерфейс TWI (I2C);

− программируемый генератор прямоугольных импульсов;

− автоматическое определение отключения основного источника питания и подключение резервного;

− потребление не более 500 нА при питании от резервной батареи питания при температуре 25 °С;

− возможность поставки в промышленном диапазоне температур от –40 °С до +85 °С .

Условное обозначение микросхемы приведено на рис. 7.6.

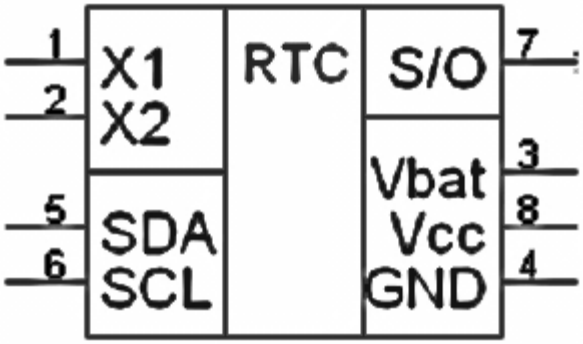


Рис. 7.6. Условное обозначение микросхемы DS1307

Микросхема DS1307 размещена в 8-контактном пластиковом корпусе типа DIP. Назначение выводов микросхемы DS1307 приведены в табл. 7.6.

Управление микросхемой осуществляется по последовательному TWI интерфейсу. Микросхема имеет 7-битный идентификатор 1101000.

Диапазон адресного пространства микросхемы приведен в табл. 7.7.

Время и календарь устанавливаются записью в соответствующие регистры. Бит 7 регистра 0 – бит остановки часов. Часы начинают работу после записи логического «0» в 7-й бит регистра 0.

Таблица 7.6

Назначение выводов микросхемы DS1307

Выводы	**Номер**	**Назначение**
VCC	8	Основной источник питания
X1, X2	1, 2	Подключение кварцевого резонатора 32,768 кГц
VBAT	3	Подключения резервного питания (+3 В)
GND	4	Общий
SDA	5	Ввод-вывод данных в последовательном формате
SCL	6	Синхронизация последовательной связи
SQW/OUT	7	Выход генератора прямоугольных импульсов/ выход программируемого ключа

Таблица 7.7

Адресное пространство

Адрес	**Бит 7**	**Бит 6**	**Бит 5**	**Бит 4**	**Бит 3**	**Бит 2**	**Бит 1**	**Бит 0**	**Функция**	**Диапазон**
0x0	CH	Десятки секунд			Секунды				Секунды	00…59
0x1	0	Десятки минут			Минуты				Минуты	00…59
0x2	0	12 24	Десятки часов PM/AM	Десятки часов	Часы				Часы	1…12 + AM/PM 0…23
0x3	0	0	0	0	0		День недели		День	01…07
0x4	0	0	Десятки даты		Дата				Дата	01…31
0x5	0	-	-	Десятки месяцев	Месяцы				Месяцы	01…12
0x6	Десятки лет				Года				Года	00…99
0x7	OUT	0	0	SQWE	0	0	RS1	RS0	Управление	-
0x8… 0x3F	-	-	-	-	-	-	-	-	ОЗУ 56x8	0x0…0xFF

7.6. Электрическая принципиальная схема

Электрическая принципиальная схема к данной лабораторной работе представлена в прил. 2. Для организации клавиатуры используются кнопки S1–S12, включенные по матричной схеме. Сигналы выбора ряда матрицы формируются на транзисторных ключах VT1, VT3, VT5 и VT7, управляемых выходами Q0–Q3 дешифратора DD3 HC138. Для управления дешифратором используются выходы порта микроконтроллера PC.0–PC.2. Для считывания состояния нажатых клавиш используются порты PC.5–PC.7. Для управления ЖКИ используются линии PF.0…PF.3, PE.2, PD.5 и PD.7 микроконтроллера, а для организации шины I2C – порты PD.0, PD.1.

Практическая часть

7.7. Контрольные вопросы

1. Как осуществляется объединение устройств по интерфейсу TWI (I2C)?
2. Каков формат пакетов, передаваемых по интерфейсу TWI?
3. Каковы условия СТАРТа и ОСТАНОВа?
4. Поясните принцип работы модуля TWI микроконтроллера.
5. Какие регистры управления шиной TWI вы знаете?
6. Опишите порядок инициализации шины.
7. Каково условное графическое обозначение микросхемы DS1307?
8. Что собой представляет адресное пространство микросхемы DS1307?
9. Приведите алгоритм передачи и приема информации микросхемой по шине TWI.

7.8. Варианты заданий

Разработать программу для учебного стенда НТЦ 31.100, позволяющую реализовать:

1. Часы реального времени с отображением времени на ДСИ.
2. Часы реального времени со срабатывающим «будильником» во время, введенное с клавиатуры, с отображением всех значений на МЖКИ («будильник» выбрать самостоятельно).
3. Часы реального времени с дополнительным выведением на МЖКИ даты, номера месяца и дня недели.
4. Программу, в которой в установленное с клавиатуры время срабатывает ДД1.
5. Программу, в которой в установленное с клавиатуры время срабатывает ДД1. Информацию выводить на МЖКИ.
6. Программу, в которой через 10 секунд после включения стенда срабатывает ДД1.

7. Программу, в которой через 6 секунд после включения стенда загораются все сегменты ДСИ.

8. Программу, в которой через 8 секунд после включения стенда на МЖКИ выводится сообщение «Тревога!».

9. Секундомер с выводом информации на МЖКИ.

10. Секундомер с выводом информации на ДСИ.

11. Секундомер с выводом информации на ДСИ, который каждые 6 секунд увеличивает счетчик событий на 1.

12. Секундомер с выводом информации на МЖКИ, который каждые 8 секунд увеличивает счетчик событий на 1.

13. Таймер обратного отсчета с 15 секунд до нуля с отображением информации на ДСИ.

14. Таймер обратного отсчета с 15 секунд до нуля с отображением информации на МЖКИ.

15. Таймер обратного отсчета с заданием времени отсчета с клавиатуры и выводом информации на МЖКИ. При обнулении таймера выдать сообщение «Время вышло».

7.9. Пример выполнения лабораторной работы

Задача

Реализовать часы с выводом на МЖКИ.

Решение

```
.include "m128def.inc"
.cseg
        .org    0000
        jmp     Init
Init:           ;Инициализация указателя вершины стека
        ldi     R16,0x00    ;Формируем стек, начиная с ячейки памяти 0x400
        out     SPL,R16
        ldi     R17,0x04
        out     SPH,R17
        sbi     DDRD,5      ; R/W
        sbi     DDRD,7      ; RS
        sbi     DDRE,2      ; E
        ldi     R16,0b00001111  ; DB7..DB4
        sts     DDRF,R16
        cbi     PortD,5     ; R/W = 0 (Write)
;Задержка
        ldi     R17,0
        dec     R17
        brne    PC-1
        dec     R18
```

```
        brne    PC-3
        cbi     PortD,7         ;RS = 0
        ldi     R16,0b0011
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        dec     R17
        brne    PC-1
        dec     R18
        brne    PC-3
        ldi     R16,0b0011
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        dec     R17
        brne        PC-1
        ldi     R16,0b0011
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b0010      ;4-битный интерфейс
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b0010      ;Подтверждение 4-битного интерфейса
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b1000      ;N = 1, F = 0 (симв. матрица 5x7 пикселей)
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Гашение дисплея
        ldi     R16,0b0000
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
```

```
        ldi     R16,0b1000      ;D = 0, C = 0, B = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Установка режима ввода
        ldi     R16,0b0000
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b0110      ;I/D = 1, S = 0
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Включение индикатора
        ldi     R16,0b0000
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        ldi     R16,0b1111      ;D = 1, C = 1, B = 1
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Регистры для работы с TWI
        clr     r21
        clr     r22
        clr     r23
        clr     r25
;Регистры-счетчики
        clr     r17
        clr     r18
        clr     r19
;Регистр для работы с дисплеем:
        clr     r16
;Инициализация и2с
        ldi     r25,5           ;Задаем скорость связи по шине
        sts     TWBR,r25
;Загружаем коэффициент деления частоты генератора в регистр скор. связи
        ldi     r25,(0<<TWPS1)|(0<<TWPS0)
;Устанавливаем значение предделения скорости связи, равное 1
```

```
        sts     TWSR,r25
;Загружаем значение предделителя в регистр состояния шины
;Задание массива чисел для вывода на дисплей
        ldi     r26,0x00
        ldi     r27,0x03
;Задание массива чисел для записи после нажатия клавиши
        ldi     r28,0x00
        ldi     r29,0x03
;Очищаем область памяти массива
        ldi     r16,0x00
        st      x+,r16
        andi    r26,7
        brne    pc-3
        clr     r30
        clr     r31
rcall   DS_Start                ;Вызываем старт часов
start:
;Установка курсора в левом верхнем углу 0x80
        cbi     PortD,7         ;RS = 0 – установка дисплея на прием команд
        ldi     R16,0x8         ;Отправляем старшую тетраду команды
        sts     PortF,R16       ;Отправляем ее
        sbi     PortE,2         ;Активируем прием дисплеем
        call    delay           ;Ждем
        cbi     PortE,2         ;Закрываем прием
        ldi     R16,0x0         ;Загружаем младшую тетраду команды
        sts     PortF,R16       ;Отправляем
        sbi     PortE,2         ;Активируем прием
        call    delay           ;Ждем
        cbi     PortE,2         ;Завершаем прием
;Запись символа "В" 0x42
        ldi     r16,0x42
        call    lcd_print_symbol
;Запись символа "р" 0x70
        ldi     r16,70
        call    lcd_print_symbol
;Запись символа "е" 0x65
        ldi     r16,0x65
        call    lcd_print_symbol
;Запись символа "м" 0xBC
        ldi     r16,0xBC
        call    lcd_print_symbol
;Запись символа "я" 0xC7
        ldi     r16,0xC7
        call    lcd_print_symbol
```

```
;Запись символа " " 0x20
        ldi     r16,0x20
        call    lcd_print_symbol
;Запись символа " " 0x20
        ldi     r16,0x20
        call    lcd_print_symbol
        rcallDS_get_time        ;Считываем время с часов
;Запись первого символа значения часов
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r23
        andi    r16,0xf0
        swap    r16
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Запись второго символа значения часов
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r23
        andi    r16,0x0f
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Запись символа ":" 0x3A
        ldi     r16,0x3A
        call    lcd_print_symbol
;Запись первого символа значения минут
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r22
        andi    r16,0xf0
        swap    r16
```

```
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Запись второго символа значения минут
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r22
        andi    r16,0x0f
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Запись символа ":" 0x3A
        ldi     r16,0x3A
        call    lcd_print_symbol
;Запись первого символа значения секунд
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r21
        andi    r16,0xf0
        swap    r16
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
;Запись второго символа значения секунд
        ldi     R16,0x3
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
        mov     r16,r21
        andi    r16,0x0f
        sts     PortF,R16
        sbi     PortE,2
        call    delay
        cbi     PortE,2
```

```
;Запись символа " " 0x20
        ldi     r16,0x20
        call    lcd_print_symbol
jmpstart                        ;Возврат на начало программы
DS_Start:                       ;Процедура запускает ход часов
        call    i2c_start       ;Отправляем стартовую посылку
        ldi     r25,0b11010000  ;Адрес микросхемы
        call    i2c_send        ;Отправляем его в шину
        ldi     r25,0x00        ;Адрес ячейки памяти
        call    i2c_send        ;Отправляем в шину
        call    i2c_start       ;Повторная отправка стартовой посылки
        ldi     r25,0b11010001  ;Адрес микросхемы с битом чтения
        call    i2c_send        ;Отправляем в шину
        call    i2c_recive_last ;Читаем нулевой байт
        cbr     r25,0b10000000  ;Сбрасываем 7-й бит
        push    r25             ;Сохраняем в стек
        call    i2c_stop        ;Отправляем стоповую посылку
        call    i2c_start       ;Отправляем стартовую посылку
        ldi     r25,0b11010000  ;Адрес микросхемы + бит записи
        call    i2c_send        ;Отправляем в шину
        ldi     r25,0           ;Адрес ячейки памяти
        call    i2c_send        ;Отправляем в шину
        pop     r25             ;Загружаем из стека
        call    i2c_send        ;Отправляем в шину
        call    i2c_stop        ;Завершаем
ret
DS_get_time:
;Считывание времени из часов DS1307
;Секунды будут записаны в R21, минуты в R22, часы в R23
Call i2c_start                  ;Отправляем стартовую посылку
        ldi     r25,0b11010000  ;Адрес микросхемы
        call    i2c_send        ;Отправляем в шину
        ldi     r25,0x00        ;Адрес ячейки памяти
        call    i2c_send        ;Отправляем в шину
        call    i2c_start       ;Повторная отправка стартовой посылки
        ldi     r25,0b11010001  ;Адрес микросхемы с битом чтения
        call    i2c_send        ;Отправляем в шину
        call    i2c_recive      ;Читаем секунды
        cbr     r25,0b10000000  ;Сбрасываем 7-й бит
        mov     r21,r25         ;Копируем секунды в R21
        call    i2c_recive      ;Читаем минуты
        mov     r22,r25         ;Копируем минуты в R22
        rcall   i2c_recive_last ;Читаем часы
        mov     r23,r25         ;Копируем часы в R23
```

```
        call    i2c_stop            ;Отправляем стоповую посылку
ret
i2c_stop:                           ;Отправляет стоповую посылку
        ldi     r25, (1<<TWINT)|(1<<TWEN)|(1<<TWSTO)
;Устанавливаем бит прерывания, бит разрешения работы, бит "Стоп"
        sts     TWCR, r25           ;Загружаем установленные биты в регистр
                                    ;управления
ret
i2c_start:                          ;Передача стартовой посылки
        ldi     r25, (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)
;Устанавливаем биты прерыв., условия "старт" и разреш. работы шины
        sts     TWCR, r25
;Отправляем установленные биты в регистр управления работой шины
        wait1:                      ;Ждем подтверждения
        lds     r25,TWCR
;Загружаем значение регистра управления шиной
        sbrs    r25,TWINT           ;Проверяем, установлен ли флаг прерываний
        jmp     wait1               ;Если нет, прыгаем обратно на wait1, если да, то
                                    ;выходим из подпрограммы
ret
i2c_recive_last:            ;Считать последний байт и записать в R25
        ldi     r25, (1<<TWINT|1<<TWEN)
;Устанавливаем биты прерывания и бит разрешения работы
        sts     TWCR, r25
;Загружаем установленные биты в регистр управления
wait3:                              ;Ждем получения
        lds     r25,TWCR            ;Загружаем в R25 значение регистра управления
                                    ;шиной
        sbrs    r25,TWINT           ;Проверяем, установлен ли бит прерываний
        jmp     wait3               ;Если нет – ждем дальше
        lds     r25,TWDR            ;Если да, считываем в R25 пришедшие данные
ret                                 ;Выходим из подпрограммы
i2c_recive:                         ;Считывает байт и записывает его в R25
        ldi     r25, (1<<TWINT|1<<TWEN|1<<TWEA)
;Устанавливаем биты прерыв., разреш. работы и бит условия "Старт"
        sts     TWCR, r25           ;Загружаем эти биты в регистр управления
                                    ;шиной
wait4:                              ;Ждем получения
        lds     r25,TWCR            ;Загружаем значение регистра управления шиной
        sbrs    r25,TWINT           ;Проверяем, установлен ли бит прерываний
        jmp     wait4               ;Если нет – ждем дальше
        lds     r25,TWDR            ;Если да – загружаем в R25 значение регистра
                                    ;данных шины
ret                                 ;Выход из подпрограммы
```

```
i2c_send:                   ;Передача байта из регистра R25
        sts TWDR,r25        ;Загружаем в регистр данных значение R25
        ldi r25, (1<<TWINT)|(1<<TWEN)
;Устанавливаем биты прерываний и разрешения работы
        Sts TWCR, r25
;Загружаем биты в регистр управления работой шины
wait2:                      ;Ждем подтверждения
        lds r25,TWCR        ;Проверяем значение регистра управления
                            ;работой шины
        sbrs   r25,TWINT    ;Если флаг прерываний установлен, то передача
                            ;завершена
        rjmp   wait2        ;Если нет, ждем дальше
ret
lcd_print_symbol:           ;Запись символа на ЖК дисплей
        sbi    PortD,7      ;RS = 1 – установка дисплея на прием символов
        mov    r17,r16      ;Копируем выводимый символ в R16
        andi   r16,0xF0     ;Удаляем младшую тетраду из R16
        swap   r16          ;Меняем местами тетрады
        sts    PortF,R16    ;Отправляем эту тетраду на дисплей
        sbi    PortE,2      ;Включаем прием
        call   delay
        cbi    PortE,2      ;Выключаем прием
        mov    r17,r16      ;Снова копируем в R16
        andi   r16,0x0F     ;Удаляем старшую тетраду – остается только
                            ;младшая
        sts    PortF,R16    ;Отправляем ее на дисплей
        sbi    PortE,2      ;Включаем прием
        call   delay
        cbi    PortE,2      ;Выключаем прием
ret                         ;Выход из подпрограммы
delay:                      ;Блок задержки
clr    r17                  ;Очищаем используемый регистр
dec    r17                  ;Уменьшаем на единицу
brne   pc-1                 ;Если не ноль – возвращаемся на уменьшение
ret                         ;Если ноль – выход их подпрограммы
;Конец программы
```

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ЛАБОРАТОРНАЯ РАБОТА №8
ИССЛЕДОВАНИЕ ОРГАНИЗАЦИИ И РАБОТЫ ПОСЛЕДОВАТЕЛЬНОГО ПЕРИФЕРИЙНОГО ИНТЕРФЕЙСА SPI

Цель работы: ознакомиться с последовательным периферийным интерфейсом SPI, микросхемой энергонезависимой памяти, изучить схему сопряжения микросхемы энергонезависимой памяти DATAFLASH с микроконтроллером, разработать и отладить программу для обслуживания устройства.

Теоретическая часть

8.1. Приборная шина SPI

SPI – популярный интерфейс для последовательного обмена данными между микросхемами. Интерфейс SPI наряду с I2C относится к самым широко используемым интерфейсам для соединения микросхем. Изначально он был реализован компанией Motorola, а в настоящее время используется в продукции многих производителей. Его наименование является аббревиатурой от «Serial Peripheral Bus», что отражает его предназначение – шина для подключения внешних устройств. Шина SPI организована по принципу «ведущий/подчиненный».

Главным составным блоком интерфейса SPI является обычный сдвиговый регистр, сигналы синхронизации и ввода-вывода битового потока которого и образуют интерфейсные сигналы. Таким образом, протокол SPI правильнее назвать не протоколом передачи данных, а протоколом обмена данными между двумя сдвиговыми регистрами, каждый из которых одновременно выполняет и функцию приемника, и функцию передатчика. Непременным условием передачи данных по шине SPI является генерация сигнала синхронизации шины. Этот сигнал имеет право генерировать только ведущий шины и от этого сигнала полностью зависит работа подчиненного шины.

Существует три типа подключения к шине SPI, в каждом из которых участвуют четыре сигнала. Самое простое подключение, в котором участвуют только две микросхемы, показано на рис. 8.1. Здесь ведущий шины передает данные по линии MOSI синхронно со сгенерированным им же сигналом SCLK, а подчиненный захватывает переданные биты данных по определенным фронтам принятого сигнала синхронизации. Одновременно с этим подчиненный отправляет свою посылку данных. Представленную схему можно упростить исключением линии MISO, если используемая подчиненная ИС не предусматривает ответную передачу данных или в ней нет потребности. Одностороннюю передачу данных можно встретить у таких микросхем, как ЦАП, цифровые потенциометры, программируемые усилители и драйверы. Таким образом, рассматриваемый вариант подключения подчиненной ИС требует 3 или 4 линии связи. Чтобы подчиненная ИС принимала и передавала данные, кроме наличия

сигнала синхронизации, необходимо также, чтобы линия SS была переведена в низкое состояние. В противном случае подчиненная ИС будет неактивна.

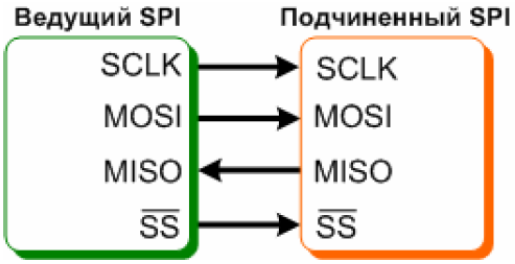


Рис. 8.1. Простейшее подключение к шине SPI

При необходимости подключения к шине SPI нескольких микросхем используется либо независимое (параллельное) подключение (рис. 8.2), либо каскадное (последовательное) (рис. 8.3). Независимое подключение более распространенное, т. к. достигается при использовании любых SPI-совместимых микросхем. Здесь все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины переводом того или иного сигнала SS в низкое состояние задает, с какой подчиненной ИС он будет обмениваться данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации подчиненных микросхем (общее число линий связи равно 3 + n, где n – количество подчиненных микросхем). Каскадное включение избавлено от этого недостатка, т. к. здесь из нескольких микросхем образуется один большой сдвиговый регистр. Для этого выход передачи данных одной ИС соединяется со входом приема данных другой, как показано на рис. 8.3. Входы выбора микросхем здесь соединены параллельно и, таким образом, общее число линий связи сохранено равным 4. Однако использование каскадного подключения возможно только в том случае, если его поддержка указана в документации на используемые микросхемы.

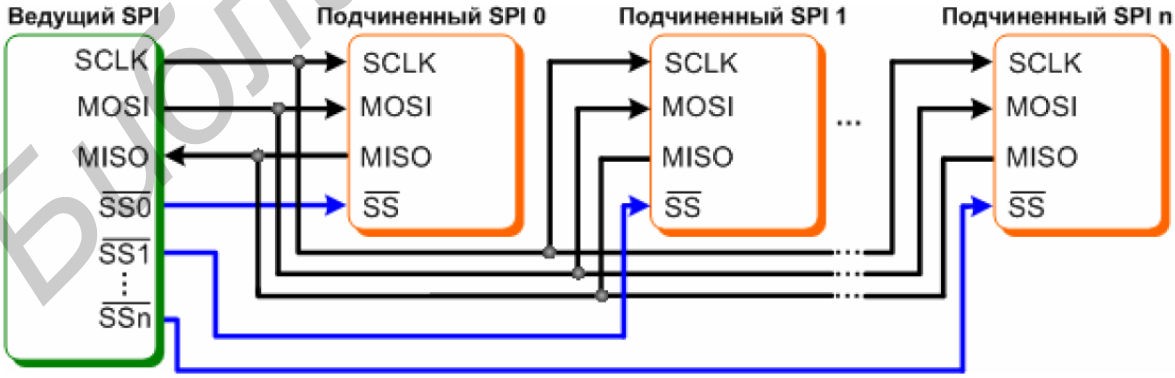


Рис. 8.2. Независимое подключение к шине SPI

Протокол передачи по интерфейсу SPI предельно прост и, по сути, идентичен логике работы сдвигового регистра, которая заключается в выполнении

операции сдвига и соответственно побитного ввода и вывода данных по определенным фронтам сигнала синхронизации. Установка данных при передаче и выборка при приеме всегда выполняются по противоположным фронтам синхронизации. Это необходимо для гарантирования выборки данных после надежного их установления.

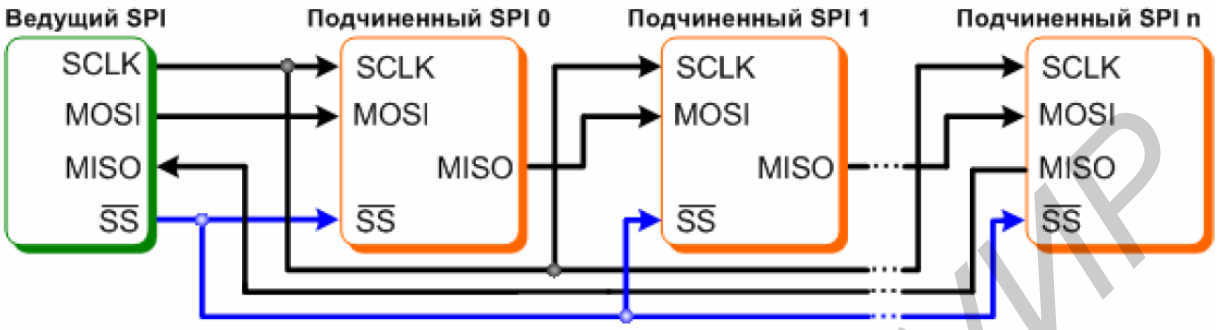


Рис. 8.3. Каскадное подключение к шине SPI

Электрические сигналы шины SPI представлены в табл. 8.1.

Таблица 8.1

Электрические сигналы шины SPI

Ведущий шины			**Подчиненный шины**		
Основное обозначение	**Альтернативное обозначение**	**Описание**	**Основное обозначение**	**Альтернативное обозначение**	**Описание**
MOSI	DO SDO OUT	Выход последовательной передачи данных	MOSI	DO SDO OUT	Выход последовательной передачи данных
MISO	DI SDI DIN	Вход последовательного приема данных	MISO	DI SDI DIN	Вход последовательного приема данных
SCLK	DCLOCK CLK SCK	Выход синхронизации передачи данных	SCLK	DCLOCK CLK SCK	Вход синхронизации передачи данных
SS	CS	Выход выбора подчиненного	SS	CS	Вход выбора подчиненного

В качестве первого фронта в цикле передачи может выступать нарастающий или падающий фронт. Всего возможно четыре варианта логики работы интерфейса SPI. Эти варианты получили название режимов SPI и описываются двумя параметрами:

1. CPOL – исходный уровень сигнала синхронизации (если CPOL = 0, то линия синхронизации до начала цикла передачи и после его окончания имеет низкий уровень, (т. е. первый фронт нарастающий, а последний – падающий), иначе, если CPOL = 1 – высокий (т. е. первый фронт падающий, а последний – нарастающий));

2. CPHA – фаза синхронизации. От этого параметра зависит, в какой последовательности выполняется установка и выборка данных (если CPHA = 0, то по переднему фронту в цикле синхронизации будет выполняться выборка данных, а затем по заднему фронту – установка данных; если же CPHA = 1, то установка данных будет выполняться по переднему фронту в цикле синхронизации, а выборка – по заднему).

Ведущая и подчиненная микросхемы, работающие в различных режимах SPI, являются несовместимыми, поэтому перед выбором подчиненных микросхем важно уточнить, какие режимы поддерживаются ведущим шины. Аппаратные модули SPI, интегрированные в микроконтроллеры, в большинстве случаев поддерживают возможность выбора любого режима SPI и, поэтому к ним возможно подключение любых подчиненных SPI-микросхем (относится только к независимому варианту подключения). Кроме того, протокол SPI в любом из режимов легко реализуется программно.

8.2. Реализация интерфейса SPI в микроконтроллерах ATMega

Отличительными особенностями интерфейса SPI (рис. 8.4) в микроконтроллерах ATmega128 являются:

– полнодуплексная, трехпроводная синхронная передача данных;
– ведущая или подчиненная работа;
– передача первым младшего или старшего бита;
– семь программируемых скоростей связи;
– флаг прерывания для индикации окончания передачи данных;
– защитный флаг при повторной записи;
– пробуждение из режима холостого хода (Idle);
– режим ведущего (мастера) SPI с удвоением скорости (CK/2).

Ведущий SPI инициирует сеанс связи подачей низкого уровня на вход SS того подчиненного устройства, с которым необходимо обмениваться данными. Оба респондента (ведущий и подчиненный) подготавливают данные к передаче в своем сдвиговом регистре, при этом на стороне ведущего генерируются также импульсы синхронизации на линии SCK. По линии MOSI всегда осуществляется передача данных от ведущего к подчиненному, а по MISO, наоборот, от подчиненного к мастеру. По окончании передачи каждого пакета

данных ведущий SPI должен засинхронизировать подчиненный путем подачи высокого уровня на линию SS (выбор подчиненного интерфейса).

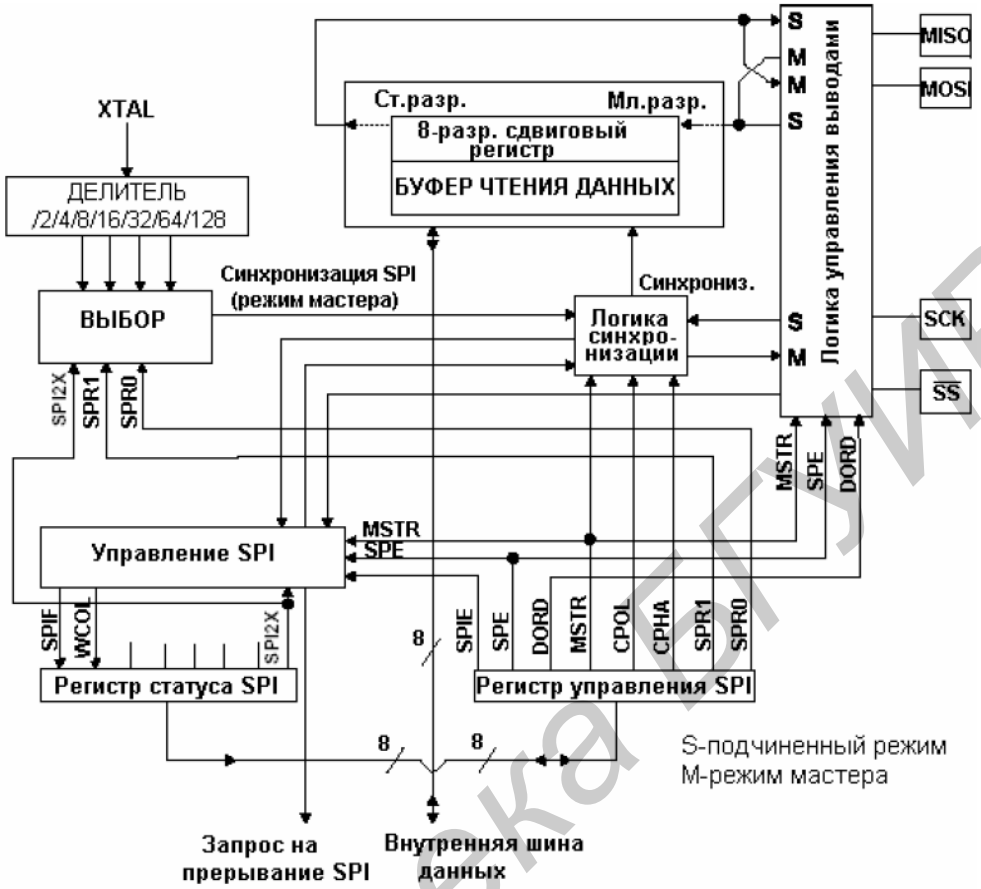


Рис. 8.4. Функциональная схема интерфейса SPI

Если SPI настроен как ведущий (мастер), то управление линией SS происходит не автоматически. Данная операция должна быть выполнена программно перед началом сеанса связи. После этого запись в регистр данных SPI инициирует генерацию синхронизации и аппаратный сдвиг восьми разрядов в подчиненное устройство. По окончании сдвига одного байта генератор синхронизации SPI останавливается, при этом устанавливая флаг окончания передачи (SPIF). Если установлен бит SPIE в регистре SPCR, то разрешается прерывание SPI и по окончании передачи байта будет генерирован запрос на прерывание. Мастер может продолжить сдвигать следующий байт, если записать его в регистр SPDR, или подать сигнал окончания пакета путем установки низкого уровня на линии SS. Последний принятый байт сохраняется в буферном регистре. В режиме подчиненного интерфейс SPI находится в состоянии ожидания, в котором MISO переводится в третье состояние до тех пор, пока на выводе SS присутствует высокий уровень. В этом состоянии программа может обновлять содержимое регистра данных SPI (SPDR), но при этом входящие импульсы синхронизации не сдвигают данные до подачи низкого уровня на вывод SS. После того как один байт был полностью сдвинут, устанавливается флаг оконча-

ния передачи SPIF. Если установлен бит разрешения прерывания SPI (SPIE) в регистре SPCR (табл. 8.2), то установка флага SPIF приводит к генерации запроса на прерывание. Подчиненный может продолжать размещать новые данные для передачи в регистр SPDR перед чтением входящих данных. Последний принятый байт хранится в буферном регистре.

В направлении передачи данных система выполнена как однобуферная, а в направлении приема используется двойная буферизация. Это означает, что передаваемые байты не могут быть записаны в регистр данных SPI прежде, чем полностью завершится цикл сдвига. Во время приема данных необходимо следить, чтобы принятая посылка была считана из регистра данных SPI прежде, чем завершится цикл входящего сдвига новой посылки. В противном случае первый байт будет потерян.

В подчиненном режиме SPI управляющая логика осуществляет выборку входящего сигнала SCK. Чтобы гарантировать корректность выборки тактового сигнала необходимо использовать частоту синхронизации SPI не более fosc/4.

Таблица 8.2

Регистр управления SPI – SPCR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Разряд 7 – SPIE: Разрешение прерывания SPI

Если установлен флаг SPIF в регистре SPSR (табл. 8.3) и установлен бит общего разрешения прерываний I в регистре SREG, то установка данного бита приведет к исполнению процедуры обработки прерывания по SPI.

Разряд 6 – SPE: Разрешение SPI

Если в SPE записать логическую «1», то разрешается работа SPI. Данный бит должен быть установлен, если необходимо использовать SPI независимо от того, в каком режиме он будет работать.

Разряд 5 – DORD: Порядок сдвига данных

Если DORD = 1, то при передаче слова данных первым передается младший разряд. Если же DORD = 0, то первым передается старший разряд.

Разряд 4 – MSTR: Выбор ведущего/подчиненного

Если в данный бит записана логическая «1», то SPI работает как ведущий (мастер), иначе (MSTR = 0) как подчиненный. Если SS настроен как вход и к нему приложен низкий уровень, когда MSTR был равен 1, то бит MSTR авто-

матически сбрасывается и устанавливается флаг прерывания SPIF в регистре SPSR. Для возобновления ведущего режима SPI пользователь должен предусмотреть программную установку бита MSTR.

Разряд 3 – CPOL: Полярность синхронизации

Если данный бит равен логической «1», то SCK имеет высокий уровень в состоянии ожидания. Если CPOL = 0, то SCK имеет низкий уровень в состоянии ожидания.

Разряд 2 – CPHA: Фаза синхронизации

Значение бита фазы синхронизации (CPHA) определяет, по какому фронту SCK происходит выборка данных: по переднему или заднему.

Разряды 1, 0 – SPR1, SPR0: Биты 1 и 0 выбора частоты синхронизации

Данные биты задают частоту синхронизации (табл. 8.4) на выводе SCK в режиме мастера. SPR1 и SPR0 не оказывают никакого влияния в режиме подчиненного.

Таблица 8.3

Регистр статуса SPI – SPSR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Имя	SPIF	WCOL	-	-	-	-	-	SPI2X
Чтение/ запись	Чт	Чт	Чт	Чт	Чт	Чт	Чт	Чт/Зп
Исходное значение	0	0	0	0	0	0	0	0

Таблица 8.4

Выбор частоты синхронизации

SPI2X	SPR1	SPR0	Частота SCK
0	0	0	fosc/4
0	0	1	fosc/16
0	1	0	fosc/64
0	1	1	fosc/128
1	0	0	fosc/2
1	0	1	fosc/8
1	1	0	fosc/32
1	1	1	fosc/64

Разряд 7– SPIF: Флаг прерывания по SPI

Флаг SPIF устанавливается по завершении последовательной передачи. Прерывание генерируется в том случае, если установлен бит SPIE в регистре SPCR и разрешены общие прерывания. Если SS настроен как вход и к нему

приложен низкий уровень, то, если SPI находился в режиме мастера, также установится флаг SPIF. SPIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно бит SPIF сбрасывается при первом чтении регистра статуса SPI с установленным флагом SPIF, а также во время доступа к регистру данных SPI (SPDR) (табл. 8.5).

Разряд 6 – WCOL: Флаг повторной записи

Бит WCOL устанавливается, если выполнена запись в регистр данных SPI (SPDR) во время передачи данных. Бит WCOL (а также бит SPIF) сбрасывается при первом чтении регистра статуса SPI с установленным WCOL, а также во время доступа к регистру данных SPI.

Разряды 5…1 – Res: зарезервированные биты

В микроконтроллере ATmega128 данные биты не используются и всегда считываются как логический «0».

Разряд 0 – SPI2X: Бит удвоения скорости SPI

Если в данный бит записать логическую «1», то скорость работы SPI (частота SCK) удвоится, если SPI находится в режиме мастера, это означает, что минимальный период SCK будет равен двум периодам синхронизации ЦПУ. Если SPI работает как подчиненный, то работа SPI гарантирована только на частоте fosc/4 или менее.

Таблица 8.5

Регистр данных SPI – SPDR

Разряд	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Чтение/ запись	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп	Чт/Зп
Исходное значение	x	x	x	x	x	x	x	x

Регистр данных SPI имеет доступ на чтение и запись и предназначен для обмена данными между файлом регистров (R0–R31) и сдвиговым регистром SPI. Запись в данный регистр инициирует передачу данных. При чтении данного регистра фактически считывается содержимое приемного буфера сдвигового регистра.

8.3. Микросхема Flash-памяти (DataFlash) AT45DB041B

Отличительными особенностями микросхемы AT45DB041B являются:

– однополярное напряжение питания 2,5…3,6 В или 2,7…3,6 В;

– совместимость с последовательным периферийным интерфейсом типа SPI;

– максимальная тактовая частота 20 МГц;

– постраничный режим программирования;

– одиночный цикл перепрограммирования (стирание плюс программирование);

– 2048 страниц основной памяти (264 байта на страницу);

– поддержка страничного и блочного режимов стирания;

– два 264-байтных буфера данных SRAM, обеспечивающих прием данных в режиме перепрограммирования энергонезависимой памяти;

– поддержка режима непрерывного считывания полного массива данных;

– поддержка приложений теневого дублирования кода;

– низкое энергопотребление: 4 мА – типичный ток в режиме активного чтения, 2 мкА – типичный потребляемый ток КМОП в режиме ожидания;

– аппаратная функция защиты данных;

– входы сигналов SI, SCK, CS (активный низкий), RESET (активный низкий) и WP (активный низкий) устойчивы к логическим уровням 5 В.

Условное обозначение микросхемы представлено на рис. 8.5, назначение выводов приведено в табл. 8.6, а структура микросхемы – на рис. 8.6.

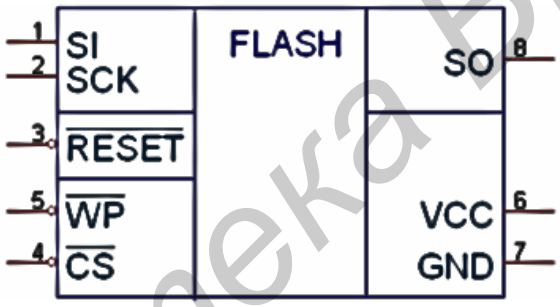


Рис. 8.5. Условное обозначение микросхемы AT45DB041B

Таблица 8.6

Назначение выводов микросхемы AT45DB041B

Наименование вывода	**Назначение**
CS (активный низкий)	Вход выбора подчиненного
SCK	Вход синхронизации
SI	Последовательный вход
SO	Последовательный выход
WP (активный низкий)	Вход защиты записи памяти
VCC	Питание
GND	Общий

4 325 376 бит памяти данной микросхемы организованы в 2048 страницах по 264 байта каждая. Кроме памяти общего назначения, микросхема также имеет два SRAM буфера данных по 264 байта. Буферы обеспечивают возможность приема данных в режиме перепрограммирования страницы основной памяти, или считывание, или запись непрерывных потоков данных. Режим эмуляции

EEPROM (с побитным или побайтным изменением) прост в применении благодаря встроенной 3-ступенчатой системе команд Read – Modify – Write. Память типа DataFlash использует последовательный интерфейс для обращения к своим данным в режиме последовательного доступа. Микросхема поддерживает SPI – режимы типа 0 и 3, функционирует с тактовыми частотами, вплоть до 20 МГц при типовом потребляемом токе в режиме активного чтения 4 мА.

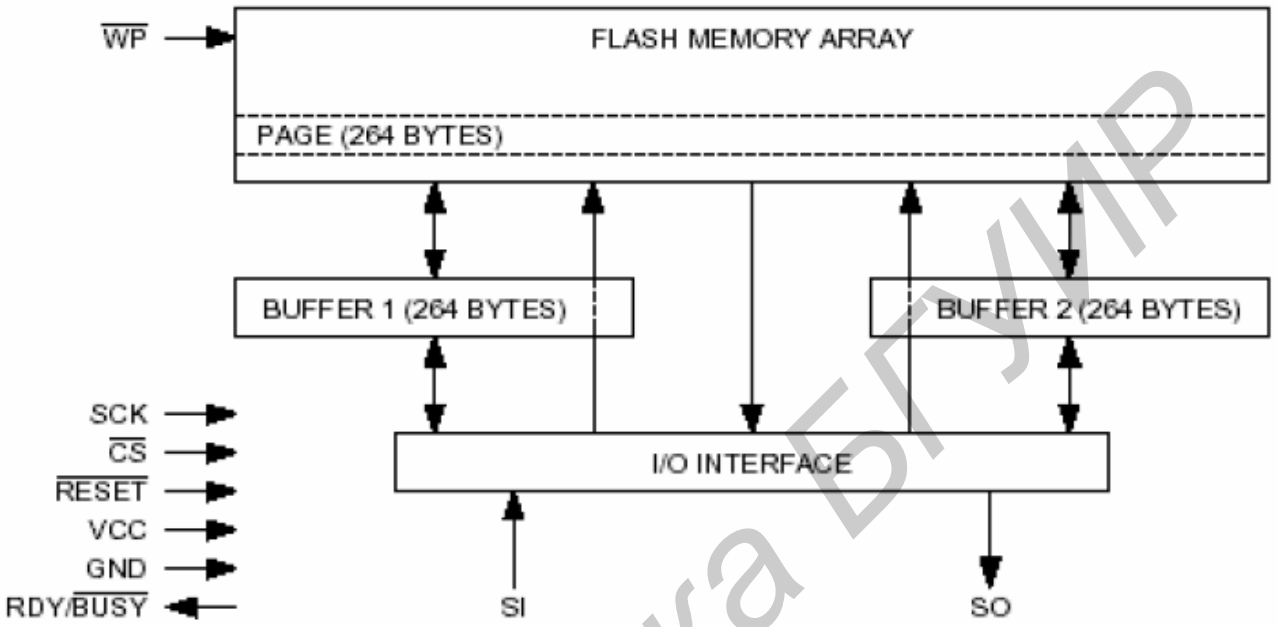


Рис. 8.6. Структура микросхемы AT45DB041B

Для обеспечения удобства внутрисистемного перепрограммирования микросхема не требует высоких входных напряжений в режиме программирования. Питание осуществляется от однополярного источника с напряжением 2,5…3,6 В или 2,7…3,6 В как в режиме программирования, так и в режиме чтения.

Матрица памяти AT45DB041B разделена на три уровня, включающих сектора, блоки и страницы. Работа устройства управляется командами, поступающими от главного процессора. Список команд и их соответствующие коды операций приведены в табл. 8.7–8.9.

Таблица 8.7

Команды чтения

Команда	**Код операции**
Непрерывное чтение массива	68h
Чтение страницы основной памяти	52h
Чтение буфера 1	54h
Чтение буфера 2	56h
Чтение регистра состояния	57h

Команда начинается по заднему фронту сигнала CS, за которым следует соответствующий 8-битный код операции и адресная ячейка требуемого буфера или основной памяти. Пока на выводе низкий уровень, переключение вывода SCK управляет загрузкой кода операции и адресной ячейки нужного буфера или основной памяти через вывод SI (последовательный вход). Все команды, адреса и данные передаются, начиная со старшего бита. Чтение происходит в режиме SPI0 или 3.

Все циклы программирования имеют встроенный контроль временных характеристик, а для проведения программирования предварительный цикл стирания не требуется.

Таблица 8.8

Команды записи и стирания

Команда	**Код операции**
Запись буфера 1	84h
Запись буфера 2	87h
Запись буфера 1 в страницу основной памяти со встроенным стиранием	83h
Запись буфера 2 в страницу основной памяти со встроенным стиранием	86h
Запись буфера 1 в страницу основной памяти без встроенного стирания	88h
Запись буфера 2 в страницу основной памяти без встроенного стирания	89h
Стирание страницы	81h
Стирание блока	50h
Запись страницы основной памяти через буфер 1	82h
Запись страницы основной памяти через буфер 2	85h

Таблица 8.9

Дополнительные команды

Команда	**Код операции**
Передача страницы основной памяти в буфер 1	53h
Передача страницы основной памяти в буфер 2	55h
Сравнение страницы основной памяти с буфером 1	60h
Сравнение страницы основной памяти с буфером 2	61h
Автоматическая перезапись страницы через буфер 1	58h
Автоматическая перезапись страницы через буфер 2	59h

Побитовая детализация адресной последовательности представлена в табл. 8.10.

Таблица 8.10

Побитовая детализация адресной последовательности

Код операции (HEX)	**Адресный байт**								**Адресный байт**								**Адресный байт**								**Доп. биты**
50	r	r	r	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	x	x	Нет
52	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
53	r	r	r	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	x	Нет
54	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
55	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	Нет
56	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
57	Нет								Нет								Нет								Нет
58	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
59	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
60	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
61	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
68	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
81	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
82	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	Нет
83	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
84	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	Нет
85	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	Нет
86	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
87	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	Нет
88	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
89	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	x	x	x	x	x	x	x	x	x	Нет
D2	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта
D4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
D6	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	B	B	B	B	B	B	B	B	B	1 байт
D7	Нет								Нет								Нет								Нет
E8	r	r	r	P	P	P	P	P	P	P	P	P	P	P	P	B	B	B	B	B	B	B	B	B	4 байта

8.4. Электрическая принципиальная схема к лабораторной работе

В электрической принципиальной схеме стенда НТЦ-31.100 (рис. 8.7) организована последовательная шина.

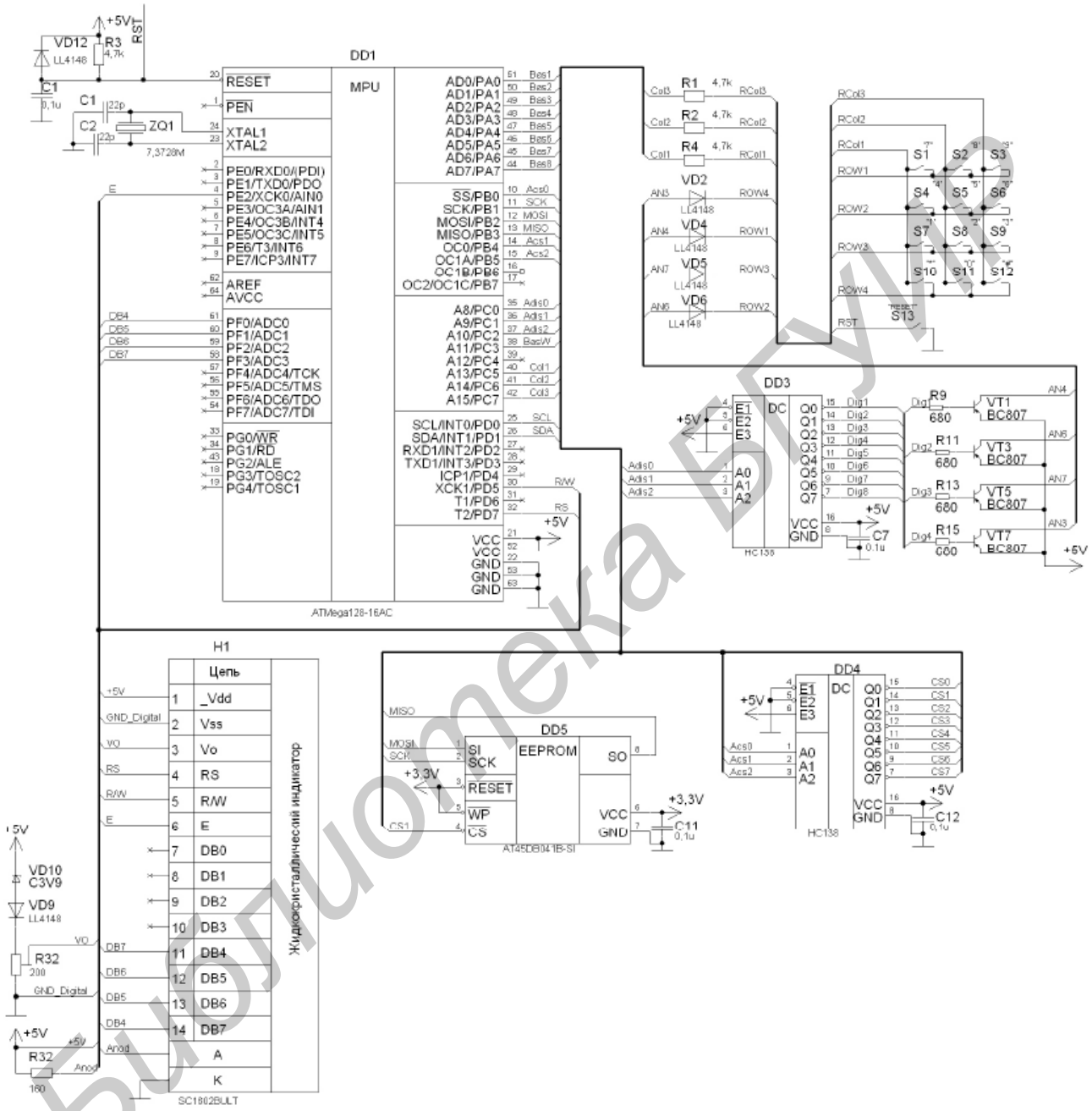


Рис. 8.7. Электрическая принципиальная схема

Для выбора конкретного устройства на шине используется набор сигналов CS0–CS7. Эти сигналы формируются на выходах дешифратора DD4 74HC138 (К1564ИД7). Для формирования кода выбираемого устройства используются порты PB.0, PB.4 и PB.5.

Для организации клавиатуры используются кнопки S1–S12, включенные по матричной схеме. Сигналы выбора ряда матрицы формируются на транзисторных ключах VT1, VT3, VT5 и VT7, управляемых выходами Q0–Q3 дешифратора DD3HC138. Для управления дешифратором используются выходы порта микроконтроллера PC.0–PC.2, для считывания состояния нажатых клавиш – порты PC.5–PC.7, для управления жидкокристаллическим индикатором – линии PF.0–PF.3, PE.2, PD.5 и PD.7 микроконтроллера.

Практическая часть

8.5. Контрольные вопросы

1. Опишите принцип работы интерфейса SPI.
2. Какие способы подключения устройств к шине SPI вы знаете?
3. Как реализован интерфейс SPI в микроконтроллерах ATMega?
4. Какие регистры управления интерфейсом вы знаете?
5. Опишите структуру микросхемы AT45DB041B.
6. Каков алгоритм отправки информации по шине SPI?
7. Каков алгоритм приема информации по шине SPI?

8.6. Варианты заданий

Напишите программу для стенда НТЦ 31.100, позволяющую:

1. Записать в основную FLASH-память микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «#» – загрузить их из FLASH-памяти с выводом на ДСИ.

2. Записать в основную FLASH-память микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «*» – загрузить их из FLASH-памяти с выводом на МЖКИ.

3.Записать в буфер 1-й микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «*» – загрузить их из FLASH-памяти с выводом на ДСИ.

4. Записать в буфер 1-й микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «#» – загрузить их из FLASH-памяти с выводом на МЖКИ.

5. Записать в буфер 2-й микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «#» – загрузить их из FLASH-памяти с выводом на ДСИ.

6. Записать в буфер 2-й микросхемы AT45DB041B вводимые с клавиатуры символы, а при нажатии клавиши «*» – загрузить их из FLASH-памяти с выводом на МЖКИ.

7. Ввести двухзначное число и вычесть из него двухзначное число, хранящееся в AT45DB041B.

8. Ввести двухзначное число и сложить его с двухзначным числом, хранящимся в AT45DB041B.

9. Ввести двухзначное число и выполнить операцию «поразрядное ИЛИ» с двухзначным числом, хранящимся в AT45DB041B.

10. Ввести двухзначное число и перемножить его с двухзначным числом, хранящимся в AT45DB041B.

11. Ввести двухзначное число и выполнить операцию «поразрядное исключающее ИЛИ» с двухзначным числом, хранящемся в AT45DB041B.

12. Ввести двухзначное число и выполнить операцию «поразрядное И» с двухзначным числом, хранящимся в AT45DB041B.

13. Ввести двухзначное число, вывести на МЖКИ предупреждение, если это число равно хранящемуся в AT45DB041B.

14. Ввести двухзначное число, вывести на МЖКИ предупреждение, если это число больше, чем хранящееся в AT45DB041B.

15. Ввести двухзначное число, вывести на МЖКИ предупреждение, если это число меньше, чем хранящееся в AT45DB041B.

8.7. Пример выполнения лабораторной работы

Задача

Разработать программу для учебного стенда НТЦ-31.100, позволяющую выводить на ЖК дисплей содержимое ячеек 100h и 101h основной памяти микросхемы AT45DB041B.

Решение

```
.include "m128def.inc"
.cseg
.org   0000
       jmp   Init
Init:
;Инициализация указателя вершины стека
       ldi   R16,0x00          ;Формируем стек, начиная с ячейки памяти 0x400
       out   SPL,R16
       ldi   R17,0x04
       out   sph,r17
       jmp   LCD_INIT          ;Инициализация работы ЖК дисплея
complete:
;Настройка SPI для работы с flash
       sbi   DDRB,0            ;Настраиваем линии порта B на передачу
       sbi   DDRB,4
       sbi   DDRB,5
;Устанавливаем нужные линии порта B для выбора микросхемы
       cbi   PORTB,0
       cbi   PORTB,4
```

```
        cbi     PORTB,5
;Установка линий MOSI и SCK на вывод
        ldi     r17,(1<<DDB2)|(1<<DDB1)    out ddrb,r17

;Перевод SPI в режим мастер, установка скорости связи fck/16
        ldi     r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
        out     SPCR,r17
start:
;Снимаем слово с FLASH-памяти и заносим его в регистры R21 и R22
        call    dataflash_read

;Вывод информации на ЖК дисплей

;Установка курсора в левом верхнем углу 0x80
        cbi     PortD,7         ;RS = 0 – установка дисплея на прием команд
        ldi     R16,0x8         ;Отправляем старшую тетраду команды
        sts     PortF,R16       ;Отправляем ее
        sbi     PortE,2         ;Активируем прием дисплеем
        clr     r14
        dec     r14
        brne    pc-1            ;Ожидаем
        cbi     PortE,2         ;Закрываем прием
        ldi     R16,0x0         ;Загружаем младшую тетраду команды
        sts     PortF,R16       ;Отправляем
        sbi     PortE,2         ;Активируем прием
        clr     r14
        dec     r14
        brne    pc-1            ;Ожидаем
        cbi     PortE,2         ;Завершаем прием
        sbi     PortD,7         ;RS = 1– установка дисплея на прием символов
;Запись символа "F" 0x46
        ldi     r20,0x46
        call    lcd_print_symbol
;Запись символа "l" 0x6C
        ldi     r20,0x6C
        call    lcd_print_symbol
;Запись символа "a" 0x61
        ldi     r20,0x61
        call    lcd_print_symbol
;Запись символа "s" 0x73
        ldi     r20,0x73
        call    lcd_print_symbol
;Запись символа "h" 0x68
        ldi     r20,0x68
```

```
        call    lcd_print_symbol
;Запись символа " " 0x20
        ldi     r20,0x20
        call    lcd_print_symbol
;Запись символа " " 0x20
        ldi     r20,0x20
        call    lcd_print_symbol
;Запись первого символа
        mov   r20,r21
        andi   r20,0xF0
        swap  r20
        call    lcd_print_number
;Запись второго символа
        mov   r20,r21
        andir20,0x0f
        call    lcd_print_number
;Запись третьего символа
        mov   r20,r22
        andir20,0xF0
        swap r20
        call    lcd_print_number
;Запись четвертого символа
        mov   r20,r22
        andi   r20,0x0f
        call    lcd_print_number
        jmp    start                ;Возвращаемся на старт

;Блок подпрограмм
dataflash_read:                     ;Операция чтения с FLASH-памяти

;Выдаем низкий сигнал на линию SC, чтобы включить устройство
        sbi     PORTB,0
        ldi     r20,0x52            ;Код команды чтения
        call    spi_send
        ldi     r20,0ч00            ;Первый байт адреса
        call    spi_send
        ldi     r20,0x01            ;Второй байт адреса
        call    spi_send
        ldi     r20,0x00            ;Третий байт адреса
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
```

```
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send

;Принимаем ответ
        call    spi_recieve
        mov     r21,r20             ;Сохраняем принятый байт

;Выдаем высокий сигнал на линию SC, чтобы отключить устройство
        cbi     PORTB,0

;Выдаем низкий сигнал на линию SC, чтобы включить устройство
        sbi     PORTB,0
        ldi     r20,0x52            ;Код команды чтения
        call    spi_send
        ldi     r20,0x00            ;Первый байт адреса
        call    spi_send
        ldi     r20,0x01            ;Второй байт адреса
        call    spi_send
        ldi     r20,0x01            ;Третий байт адреса
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
        ldi     r20,0x00            ;Незначимый байт
        call    spi_send
        call    spi_recieve         ;Принимаем ответ
        mov     r22,r20             ;Сохраняем принятый байт

;Выдаем высокий сигнал на линию SC, чтобы отключить устройство
        cbi     PORTB,0
        ret
SPI_send:                           ;Операция посылки данных в шину

;Загрузка отправляемых данных в регистр данных SPI
        out     SPDR,r20

;Перевод SPI в режим мастер, установка скорости связи fck/16
        ldi     r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
        out     SPCR,r17
        wait1:
        sbis    SPSR,SPIF           ;Ждем подтверждение передачи
```

```
        jmp   wait1
        ret
        spi_recieve:            ;Прием данных от микросхемы

;Перевод SPI в режим мастер, установка скорости связи fck/16
        ldi   r17,(1<<SPE)|(1<<MSTR)|(1<<SPR0)
        out   SPCR,r17
wait2:
        sbis  SPSR,spif

;Ожидаем подтверждение приема
        jmp   wait2
        in    r20,SPDR          ;Сохраняем в R20 принятый байт
        ret
LCD_init:                       ;Инициализация ЖК дисплея (см. лаб. работу №5)
lcd_print_symbol:

;Запись символа на ЖК дисплей. Код записан в R20 (см. лаб. работу №5)
lcd_print_number:               ;Запись символа регистра

;RS = 1 – установка дисплея на прием выводимых символов
        sbi   PortD,7
        mov   R16,r20           ;Копируем во вспомогательный регистр
        cpi   r16,0x0A          ;Сравниваем число с десятью
        brsh  A1                ;Если больше – прыгаем на запись буквы
        ldi   R16,0x3           ;Если меньше – пишем цифру (код начинается с 3)
        sts   PortF,R16
        sbi   PortE,2
        clr   r14
        dec   r14
        brne  pc-1
        cbi   PortE,2
        mov   R16,r20
        sts   PortF,R16
        sbi   PortE,2
        clr   r14
        dec   r14
        brne  pc-1
        cbi   PortE,2
        jmp   A10
        A1:                     ;Если больше – пишем букву (код начинается с 4)
        ldi   R16,0x4
        sts   PortF,R16
        sbi   PortE,2
```

```
        clr     r14
        dec     r14
        brne    pc-1
        cbi     PortE,2
        mov     R16,r20

;Вычитаем из выводимого числа 9, чтобы сформировался нужный код буквы
        subi    r16,0x09
        sts     PortF,R16
        sbi     PortE,2
        clr     r14
        dec     r14
        brne    pc-1
        cbi     PortE,2
A10:
        ret
        delay:                  ;Подпрограмма задержки
        clr     r14
        dec     r14
        brne    pc-1
        ret
;Конец программы
```

Содержание отчета по лабораторной работе: цель работы, задание, листинг программы, вывод.

ПРИЛОЖЕНИЕ 1

Основные сведения о командах микроконтроллеров семейства AVR Mega

Таблица П.1.1

Мнемоника команд

Мнемокод	Операнды	Описание	Действие	Флаги	Кол. машинных циклов
Арифметические и логические инструкции					
ADD1	Rd, Rr	Сложить два регистра	Rd <— Rd + Rr	Z,C,N,V, H	1
ADC	Rd, Rr	Сложить два регистра с переносом	Rd <— Rd + Rr + C	Z,C,N,V, H	1
ADIW	Rdl,K	Сложить слово с константой	Rdh:Rdl <— Rdh:Rdl + K	Z,C,N,V, S	2
SUB	Rd, Rr	Вычесть два регистра	Rd <— Rd - Rr	Z,C,N,V, H	1
SUBI	Rd, K	Вычесть константу из регистра	Rd <— Rd - K	Z,C,N,V, H	1
SBC	Rd, Rr	Вычесть два регистра с учетом переноса	Rd <— Rd - Rr - C	Z,C,N,V, H	1
SBCI	Rd, K	Вычесть константу из регистра с учетом переноса	Rd <— Rd - K - C	Z,C,N,V, H	1
SBIW	Rdl,K	Вычесть константу из слова	Rdh:Rdl <— Rdh:Rdl-K	Z,C,N,V, S	2
AND	Rd, Rr	Логическое И между регистрами	Rd <— Rd ╥ Rr	Z,N,V	1
ANDI	Rd, K	Логическое И между регистром и константой	Rd <— Rd ╥ K	Z,N,V	1
OR	Rd, Rr	Логическое ИЛИ между регистрами	Rd <— Rd v Rr	Z,N,V	1
ORI	Rd, K	Логическое ИЛИ между регистром и константой	Rd <— Rd v K	Z,N,V	1
EOR	Rd, Rr	Искл. ИЛИ между регистрами	Rd <— Rd E Rr	Z,N,V	1
COM	Rd	Дополнение до 0b11111111 ($FF), инверсия	Rd <— $FF - Rd	Z,C,N,V	1
NEG	Rd	Дополнение до 0b00000000 ($00)	Rd <— $00 - Rd	Z,C,N,V, H	1
SBR	Rd, K	Установка бит (бита) в регистре	Rd <— Rd v K	Z,N,V	1
CBR	Rd, K	Сброс бит (бита) в регистре	Rd <— Rd ╥ ($FF - K)	Z,N,V	1
INC	Rd	Инкремент	Rd <— Rd + 1	Z,N,V	1
DEC	Rd	Декремент	Rd <— Rd - 1	Z,N,V	1
TST	Rd	Проверка на ноль или минус	Rd <— Rd . Rd	Z,N,V	1
CLR	Rd	Сброс регистра	Rd <— Rd E Rd	Z,N,V	1
SER	Rd	Установка регистра	Rd <— $FF	Нет	1
MUL	Rd, Rr	Умножение без знака	R1:R0 <— RdxRr	Z, C	2
MULS	Rd, Rr	Умножение со знаком	R1:R0 <— RdxRr	Z, C	2
MULSU	Rd, Rr	Умножение знакового с беззнаковым числом	R1:R0 <— Rd x Rr	Z, C	2
FMUL	Rd. Rr	Дробное умножение без знака	R1:R0 <— (RdxRr) << 1	Z, C	2
FMULS	Rd, Rr	Дробное умножение со знаком	R1:R0 <— (RdxRr) << 1	Z, C	2
FMULSU	Rd, Rr	Дробное умножение знакового с беззнаковым числом	R1:R0 <— (RdxRr) << 1	Z, C	2

Продолжение табл. П.1.1

Инструкции перехода					
RJMP	k	Относительный переход	PC <— PC + k +1	Нет	2
IJMP		Косвенный переход по указателю (Z)	PC <— Z	Нет	2
JMP	k	Безусловный переход	PC <— k	Нет	3
RCALL	k	Относительный вызов процедуры	PC <— PC + k + 1	Нет	3
ICALL		Косвенный вызов процедуры по указателю (Z)	PC <— Z	Нет	3
CALL	k	Безусловный вызов процедуры	PC <— k	Нет	4
RET		Возврат из подпрограммы	PC <— STACK	Нет	4
RETI		Возврат из прерывания	PC <— STACK	I	4
CPSE	Rd,Rr	Сравнение и пропуск, если равно if (Rd = Rr)	PC <— PC + 2 или 3	Нет	1/2/3
CP	Rd,Rr	Сравнение	Rd-Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Сравнение с учетом переноса	Rd - Rr-C	Z, N,V,C,H	1
CPI	Rd,K	Сравнение регистра с константой	Rd-K	Z, N,V,C,H	1
SBRC	Rr,b	Пропуск, если бит в регистре сброшен	if (Rr(b)=O)PC <— PC + 2 или 3	Нет	1 /2/3
SBRS	Rr, b	Пропуск, если бит в регистре установлен	if (Rr(b)=1)PC <— PC+ 2 или 3	Нет	1/2/3
SBIC	P, b	Пропуск, если бит в регистре ввода-вывода сброшен	if (P(b)=O)PC <— PC + 2 или 3	Нет	1 /2/3
SBIS	P, b	Пропуск, если бит в регистре ввода-вывода установлен	if (P(b)=1)PC <— PC + 2 или 3	Нет	1 /2/3
BRBS	s, k	Переход, если флаг состояния установлен	if (SREG(s) = 1) then PC <— PC+k + 1	Нет	1/2
BRBC	s, k	Переход, если флаг состояния сброшен	if (SREG(s) = 0) then PC <— PC+k + 1	Нет	1 /2
BREQ	k	Переход, если равно	if (Z = 1) then PC <— PC + k + 1	Нет	1 /2
BRNE	k	Переход, если не равно	if (Z = 0) then PC <— PC + k + 1	Нет	1 /2
BRCS	k	Переход, если перенос установлен	if (C = 1)then PC <— PC + k+ 1	Нет	1 /2
BRCC	k	Переход, если перенос сброшен	if (C = 0) then PC <— PC + k + 1	Нет	1 /2
BRSH	k	Переход, если больше или равно	if (C = 0) then PC <— PC + k + 1	Нет	1 /2
BRLO	k	Переход, если меньше	if (C = 1) then PC <— PC + k+ 1	Нет	1 /2
BRMI	k	Переход, если минус	if (N = 1)then PC <— PC + k + 1	Нет	1 /2
BRPL	k	Переход, если плюс	if (N = 0) then PC <— PC + k + 1	Нет	1 /2
BRGE	k	Переход, если больше или равно с учетом знака	if (N e V= 0) then PC <— PC + k + 1	Нет	1 /2
BRLT	k	Переход, если меньше нуля с учетом знака	if (N e V= 1) then PC <— PC + k + 1	Нет	1 /2
BRHS	k	Переход, если флаг H установлен	if (H = 1)then PC <— PC + k + 1	Нет	1 /2
BRHC	k	Переход, если флаг H сброшен	if (H = 0) then PC <— PC + k + 1	Нет	1 /2

BRTS	k	Переход, если флаг T установлен	if (T = 1)then PC <— PC + k +1	Нет	1 /2
BRTC	k	Переход, если флаг T сброшен	if (T = 0) then PC <— PC + k + 1	Нет	1 /2
BRVS	k	Переход, если флаг V установлен	if (V = 1)then PC <— PC + k+ 1	Нет	1 /2
BRVC	k	Переход, если флаг V сброшен	if (V = 0) then PC <— PC + k + 1	Нет	1 /2
BRIE	k	Переход, если прерывания разрешены	if (I = 1)then PC <— PC + k + 1	Нет	1 /2
BRID	k	Переход, если прерывания запрещены	if (I = 0) then PC <— PC + k + 1	Нет	1 /2
Инструкции передачи данных					
MOV	Rd, Rr	Запись из регистра в регистр	Rd <— Rr	Нет	1
MOVW	Rd, Rr	Перезапись слова между регистрами	Rd+1:Rd <— Rr+1:Rr	Нет	1
LDI	Rd, K	Запись константы в регистр	Rd <— K	Нет	1
LD	Rd, X	Косвенное считывание из памяти в регистр	Rd <— (X)	Нет	2
LD	Rd, X+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (X), X <— X + 1	Нет	2
LD	Rd,-X	Предварительный декремент, а затем косвенное считывание из памяти в регистр	X <— X - 1, Rd <— (X)	Нет	2
LD	Rd,Y	Косвенное считывание из памяти в регистр	Rd <— (Y)	Нет	2
LD	Rd, Y+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Y), Y <— Y + 1	Нет	2
LD	Rd,-Y	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Y <— Y - 1, Rd <— (Y)	Нет	2
LDD	Rd,Y+q	Косвенное считывание из памяти в регистр со смещением	Rd <— (Y + q)	Нет	2
LD	Rd, Z	Косвенное считывание из памяти в регистр	Rd <— (Z)	Нет	2
LD	Rd, Z+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Z),Z <— Z+1	Нет	2
LD	Rd,-Z	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Z <— Z - 1, Rd <— (Z)	Нет	2
LDD	Rd, Z+q	Косвенное считывание из памяти в регистр со смещением	Rd <— (Z + q)	Нет	2
LDS	Rd, k	Непосредственное чтение из ОЗУ в регистр	Rd <— (k)	Нет	2
ST	X, Rr	Косвенная запись	(X) <— Rr	Нет	2
ST	X+, Rr	Косвенная запись и послед. инкремент	(X) <— Rr, X <— X + 1	Нет	2
ST	-X, Rr	Предв. декремент и косвенная запись	X <— X - 1, (X) <— Rr	Нет	2

ST	Y+, Rr	Косвенная запись и послед. инкремент	(Y) <— Rr, Y <— Y + 1	Нет	2
ST	-Y, Rr	Предв. декремент и косвенная запись	Y <— Y - 1, (Y) <— Rr	Нет	2
STD	Y+q,Rr	Косвенная запись со смещением	(Y + q) <— Rr	Нет	2
ST	Z, Rr	Косвенная запись	(Z) <— Rr	Нет	2
ST	Z+, Rr	Косвенная запись и послед. инкремент	(Z) <— Rr, Z <— Z + 1	Нет	2
ST	-Z. Rr	Предв. декремент и косвенная запись	Z <— Z - 1, (Z) <— Rr	Нет	2
STD	Z+q,Rr	Косвенная запись со смещением	(Z + q) <— Rr	Нет	2
STS	k, Rr	Непосредственная запись в ОЗУ	(k) <— Rr	Нет	2
LPM		Чтение из памяти программ	R0 <— (Z)	Нет	3
LPM	Rd, Z	Чтение из памяти программ	Rd <— (Z)	Нет	3
LPM	Rd, Z+	Чтение из памяти программ и последующий инкремент	Rd <— (Z), Z <— Z+1	Нет	3
ELPM		Расширенное чтение из памяти программ	R0 <— (RAMPZ:Z)	Нет	3
ELPM	Rd, Z	Расширенное чтение из памяти программ	Rd <— (RAMPZ:Z)	Нет	3
ELPM	Rd, Z+	Расширенное чтение из памяти программ и последующий инкремент	Rd <— (RAMPZ:Z), RAMPZ:Z <— RAMPZ:Z+1	Нет	3
SPM		Запись в память программ	(Z) <— R1 :R0	Нет	-
IN	Rd, P	Считывание из порта ввода-вывода в регистр	Rd <— P	Нет	1
OUT	P, Rr	Запись из регистра в порт ввода-вывода	P <— Rd	Нет	1
PUSH	Rr	Помещение содержимого регистра в стек	STACK <— Rr	Нет	2
POP	Rd	Извлечение из стека в регистр	Rd <— STACK	Нет	2
Битовые инструкции и инструкции тестирования бит					
SBI	P,b	Установка бита в регистре ввода-вывода	I/O(P,b) <— 1	Нет	2
CBI	P,b	Сброс бита в регистре ввода-вывода	I/O(P,b) <— 0	Нет	2
LSL	Rd	Логический сдвиг влево	Rd(n+1) <— Rd(n), Rd(0) <— 0	Z,C,N,V	1
LSR	Rd	Логический сдвиг вправо	Rd(n) <— Rd(n+1), Rd(7) <— 0	Z,C,N,V	1
ROL	Rd	Вращение влево через перенос	Rd(0)<— C,Rd(n+1)<— Rd(n),C <— Rd(7)	Z,C,N,V	1
ROR	Rd	Вращение вправо через перенос	Rd(7) <— C,Rd(n)<— Rd(n+1),C <— Rd(0)	Z,C,N,V	1
ASR	Rd	Арифметический сдвиг вправо	Rd(n)<— Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Обмен тетрадами	Rd(3..0) <— Rd(7..4), Rd(7..4) <— Rd(3..0)	Нет	1

Окончание табл. П.1.1

BSET	s	Установка флага регистра SREG	SREG(s) <— 1	SREG(s)	1
BCLR	s	Сброс флага регистра SREG	SREG(s) <— 0	SREG(s)	1
BST	Rr, b	Запись бита регистра в T	T <— Rr(b)	T	1
BLD	Rd, b	Чтение из T в бит регистра	Rd(b) <— T	Нет	1
SEC		Установка переноса	C <— 1	C	1
CLC		Сброс переноса	C <— 0	C	1
SEN		Установка флага N	N <— 1	N	1
CLN		Сброс флага N	N <— 0	N	1
SEZ		Установка флага нуля Z	Z <— 1	Z	1
CLZ		Сброс флага нуля Z	Z <— 0	Z	1
SEI		Общее разрешение прерываний	I<— 1	I	1
CLI		Общий запрет прерываний	I <— 0	I	1
SES		Установка флага S	S <— 1	S	1
CLS		Сброс флага S	S <— 0	S	1
SEV		Установка флага V в регистре SREG	V <— 1	V	1
CLV		Сброс флага V в регистре SREG	V <— 0	V	1
SET		Установка флага T в регистре SREG	T <— 1	T	1
CLT		Сброс флага T в регистре SREG	T <— 0	T	1
SEH		Установка флага H в регистре SREG	H <— 1	H	1
CLH		Сброс флага H в регистре SREG	H <— 0	H	1
Инструкции управления микроконтроллером					
NOP		Нет операции		Нет	1
SLEEP		Перевод в режим сна	(см. подробное описание режима сна)	Нет	1
WDR		Сброс сторожевого таймера	(см. подробное описание сторожевого таймера)	Нет	1
BREAK		Прерывание	Только для встроенной отладки	Нет	-

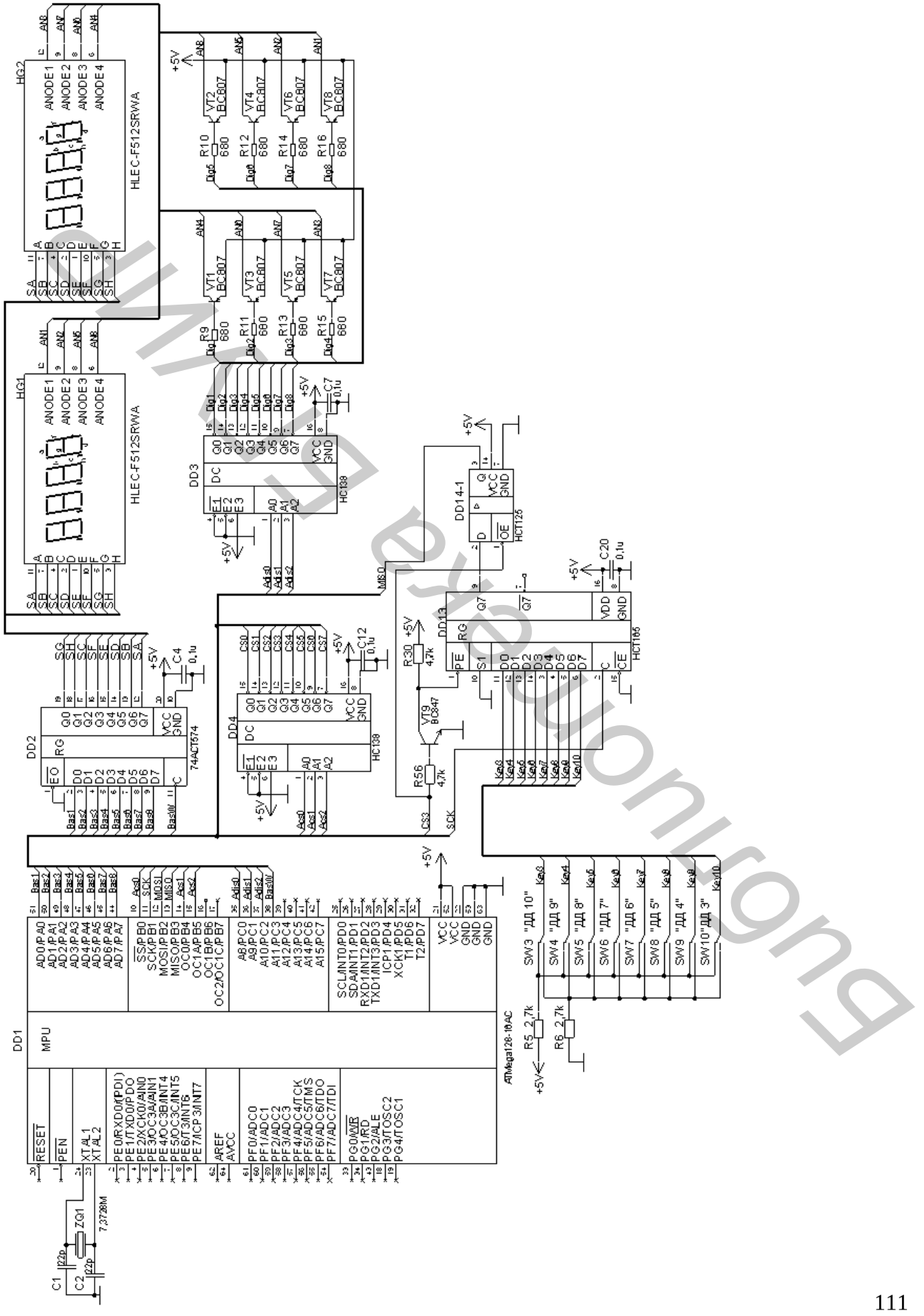
Таблица П.1.2

Таблица обозначений

Обозначение, символ	Описание
SREG	Регистр состояния микроконтроллера
C	Флаг переноса (0-й бит регистра SREG)
Z	Флаг нуля (1-й бит регистра SREG)
N	Флаг отрицательного значения (2-й бит регистра SREG)
V	Флаг переполнения дополнительного кода (3-й бит регистра SREG)
S	Флаг знака (4-й бит регистра SREG); S = N ⊕ V
H	Флаг половинного переноса (5-й бит регистра SREG)
T	Пользовательский флаг (6-й бит регистра SREG)
I	Флаг общего разрешения прерываний (7-й бит регистра SREG)
	Регистры и операнды
Rd	Регистр-приемник (иногда также регистр-источник) в регистровом файле
Rr	Регистр-источник в регистровом файле
K	Константа (данные)
к	Адрес — константа
b	Номер бита РОН или РВВ @...7)
s	Номер бита регистра состояния SREG @...7)
X, Y, Z	Индексные регистры (X = R27:R26, Y = R29:R28, Z = R31:R30)
I/O	Регистр ввода/вывода
A	Адрес в пространстве ввода/вывода
q	Смещение при относительной косвенной адресации (F-битное значение)
.	Разделитель между названием (адресом) регистра и номером бита
[XX]	Содержимое ячейки памяти данных по адресу XX
{XX}	Содержимое ячейки памяти программ по адресу XX
	Операции
–	Инверсия
•	Логическое И
∨	Логическое ИЛИ
⊕	Исключающее ИЛИ
	Система
PC	Счетчик команд
STACK	Текущий уровень стека
SP	Указатель стека
	Флаги
⇔	Команда воздействует на флаг
0	Флаг сбрасывается командой в 0
1	Флаг устанавливается командой в 1
-	Команда не влияет на состояние флага

ПРИЛОЖЕНИЕ 2

Схема электрическая принципиальная к лабораторной работе №4



ЛИТЕРАТУРА

1. Баранов, В. Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В. Н. Баранов. – М. : Издательский дом «Додэка-XXI», 2004. – 288 с.

2. Белов, А. В. Разработка устройств на микроконтроллерах AVR / А. В. Белов. – СПб. : Наука и техника, 2013. – 528 с.

3. Голубцов, М. С. Микроконтроллеры AVR: от простого к сложному / М. С. Голубцов. – М. : СОЛОН-Пресс, 2003. – 288 с.

4. Евстифеев, А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / А. В. Евстифеев. – М. : Издательский дом «Додэка-XXI», 2007. – 433 с.

5. Заец, Н. И. Радиолюбительские конструкции на PIC-микроконтроллерах / Н. И. Заец. – Киев : «МК-Пресс», 2008. – 336 с.

6. Кравченко, А. В. 10 практических устройств на AVR-микроконтроллерах. Кн. 3 / А. В. Кравченко. – М. : МК-Пресс, 2011. – 418 с.

7. Пустоваров, В. И. Язык Ассемблера в программировании информационных и управляющих систем / В. И. Пустоваров. – М. : «ЭНТРОП»; Киев: «ВЕК», 1997. – 304 с.

8. Ревич, Ю. В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера / Ю. В. Ревич. – 5-е изд., перераб. – СПб. : БХВ-Петербур, 2011. – 352 с.

9. Трамперт, В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров / В. Трамперт. – Киев : МК-Пресс, 2006. – 208 с.

Св. план 2014, поз. 2

Учебное издание

Логин Владимир Михайлович
Цырельчук Игорь Николаевич
Ролич Олег Чеславович

ИНТЕЛЛЕКТУАЛЬНЫЕ ЭЛЕКТРОННЫЕ СИСТЕМЫ БЕЗОПАСНОСТИ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

В 2-х частях

Часть 1

МИКРОКОНТРОЛЛЕРЫ СЕМЕЙСТВА AVR

ПОСОБИЕ

Редактор *М. А. Зайцева*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *В. М. Задоля*

Подписано в печать 10.03.2014. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. 6,74. Уч.-изд. л. 6,5. Тираж 100 экз. Заказ 453.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
ЛИ №02330/0494371 от 16.03.2009. ЛП №02330/0494175 от 03.04.2009.
220013, Минск, П. Бровки, 6