681.385.5(07)
М597

№ 3589

Министерство образования Российской Федерации
Государственное образовательное учреждение высшего профессионального образования
Таганрогский государственный радиотехнический университет

Ю.И.ИВАНОВ
В.Я.ЮГАЙ

МИКРОПРОЦЕССОРНЫЕ УСТРОЙСТВА СИСТЕМ УПРАВЛЕНИЯ

МЕТОДИЧЕСКОЕ РУКОВОДСТВО
К ЛАБОРАТОРНЫМ РАБОТАМ

Таганрог 2004

УДК 681.3.01(031)

Ю.И.Иванов, В.Я.Югай. Микропроцессорные устройства систем управления: Методическое руководство к лабораторным работам.–Таган-рог: Изд-во ТРТУ, 2004.– 49 с.

Методическое руководство предназначено для студентов направления 550200 «Автоматизация и управление». Цикл лабораторных работ может использоваться для практического освоения программно-аппаратных ресурсов микроконтроллеров и получения навыков практической реализации типовых функций в системах управления и может быть полезным при изучении курсов «Микропроцессорные устройства систем управления», «Электронные устройства автоматики», «Технические средства автоматизации и управления».

Ил. 7.

Рецензент:

Д-р техн. наук, профессор кафедры ВТ ТРТУ Золотовский В.Е.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Среди современных электронных компонентов, которые применяются для реализации различных алгоритмов управления, наибольший интерес представляют микроконтроллеры. В настоящее время микроконтроллеры выпускаются многими фирмами-производителями интегральных схем и являются массовыми, относительно недорогими и доступными изделиями. Интегрируя на одном кристалле высокопроизводительный процессор, память и периферийные устройства, микроконтроллеры позволяют с минимальными затратами реализовать системы управления различными объектами и процессами. Благодаря этому они находят широкое применение при решении разнообразных задач в промышленной автоматике, бытовой технике, контрольно-измерительной технике, аппаратуре связи и т.п.

В настоящее время существует большое количество микроконтроллеров с различным набором программно-аппаратных ресурсов и с определенными, соответствующими этим ресурсам, достоинствами и недостатками. Среди основных производителей можно выделить такие фирмы, как Intel, Dallas Semiconductor, Philips Semiconductor, Atmel, Motorola, Fujitsu, Microchip.

Для многих прикладных задач оптимально использование микроконтроллеров семейства **AVR** фирмы **Atmel**. Они являются эффективным инструментом для создания современных, высокопроизводительных, экономичных и многоцелевых устройств управления [1]. Соотношение "цена - производительность - энергопотребление" для AVR является одним из лучших на мировом рынке 8-разрядных микроконтроллеров.

Важнейшее достоинство микроконтроллеров семейства AVR – высокие функциональные возможности при относительно низких затратах, поэтому их применение целесообразно как при реализации сложных алгоритмов управления, так и при решении простых задач. Высокая эффективность микроконтроллеров AVR обеспечивается развитой системой команд, выполняющихся, как правило, за один рабочий такт, аппаратной реализацией многих стандартных функций (таймеры, модуляторы ШИМ, параллельные и последовательные порты ввода-вывода, компаратор, АЦП и др.) и внутрисистемным программированием, т.е. возможностью записи программ и данных в ПЗУ микроконтроллера непосредственно в схеме работающего устройства. Возможность многократного программирования обеспечивает функциональную гибкость, которая позволяет использовать одни и те же средства реализации при решении весьма разнообразных задач.

Данный цикл лабораторных работ и лабораторный стенд может использоваться при изучении курсов «Микропроцессорные устройства систем управления», «Электронные устройства автоматики», «Технические средства автоматизации и управления» и др. При соответствующей постановке задач в

лабораторном цикле могут рассматриваться вопросы реализации разнообразных типовых функций.

Реализация необходимых функций микроконтроллером требует эффективного управления его программно-аппаратными средствами. Это управление и координация работы различных средств производится рабочими программами. В связи с тем, что значительная часть функций в микроконтроллере реализуется аппаратными средствами, а ассемблер обеспечивает более рациональное управление этими средствами, в лабораторных работах предполагается использование ассемблера в качестве основного языка программирования.

Для получения необходимых навыков и практического опыта работы с микроконтроллерами требуются специальные средства, включающие инструментальные средства разработки программ и программирования микроконтроллеров, а также устройства для аппаратной реализации и экспериментальной проверки корректности и эффективности используемых алгоритмов работы. Цикл лабораторных работ предназначен для изучения программно-аппаратных ресурсов микроконтроллеров семейства AVR и особенностей их применения в реализации типовых для систем управления алгоритмов. Подготовленные при выполнении данных лабораторных работ программные модули в дальнейшем могут использоваться как стандартные элементы при решении более сложных задач.

1. ЛАБОРАТОРНЫЙ СТЕНД

Подготовку текста программы микроконтроллера AVR можно выполнить на персональном компьютере с помощью инструментального пакета **AVR Studio**, запись подготовленного файла программы микроконтроллера производится из AVR Studio программатором. Особенности программно-аппаратных ресурсов микроконтроллеров AVR описаны в [1], система команд приведена в приложении и в [1]. Экспериментальная проверка алгоритмов работы невозможна без различных устройств сопряжения, управления и индикации. Решение этой задачи требует применения специального отладочного стенда, содержащего микроконтроллер и все необходимые дополнительные средства реализации стандартных функций управления.

Структурная схема лабораторного стенда, позволяющего обрабатывать и формировать реальные сигналы с возможностью физической реализации типовых алгоритмов, приведена на рис. 1.1. Лабораторный стенд предназначен для проведения циклов лабораторных работ по нескольким дисциплинам, необходимая универсальность обеспечивается следующими функциональными узлами стенда:

- микроконтроллер AT90S8535;
- программатор AVR;
- приемопередатчики стандартных последовательных портов;
- CAN-контроллер и приемопередатчик CAN-сети;
- аналоговый усилитель мощности;
- умножающий 12-разрядный цифроаналоговый преобразователь;
- управляемый источник опорного напряжения;
- индикаторы и контроллер LED индикатора;
- кнопки управления и аналоговый потенциометр;
- стабилизаторы постоянного напряжения с защитой от короткого замыкания.

На лицевой панели стенда установлены:

- цифровой шестиразрядный светодиодный индикатор (LED). Этот индикатор предназначен для отображения необходимой информации и подключен к микроконтроллеру через последовательный интерфейс SPI и контроллер LED;
- две линейки светодиодов, предназначенных для отображения состояния сигналов линий ввода-вывода портов PORTC и PORTD микроконтроллера;
- четыре управляющие кнопки, которые позволяют производить управление работой с помощью внешних прерываний микроконтроллера;

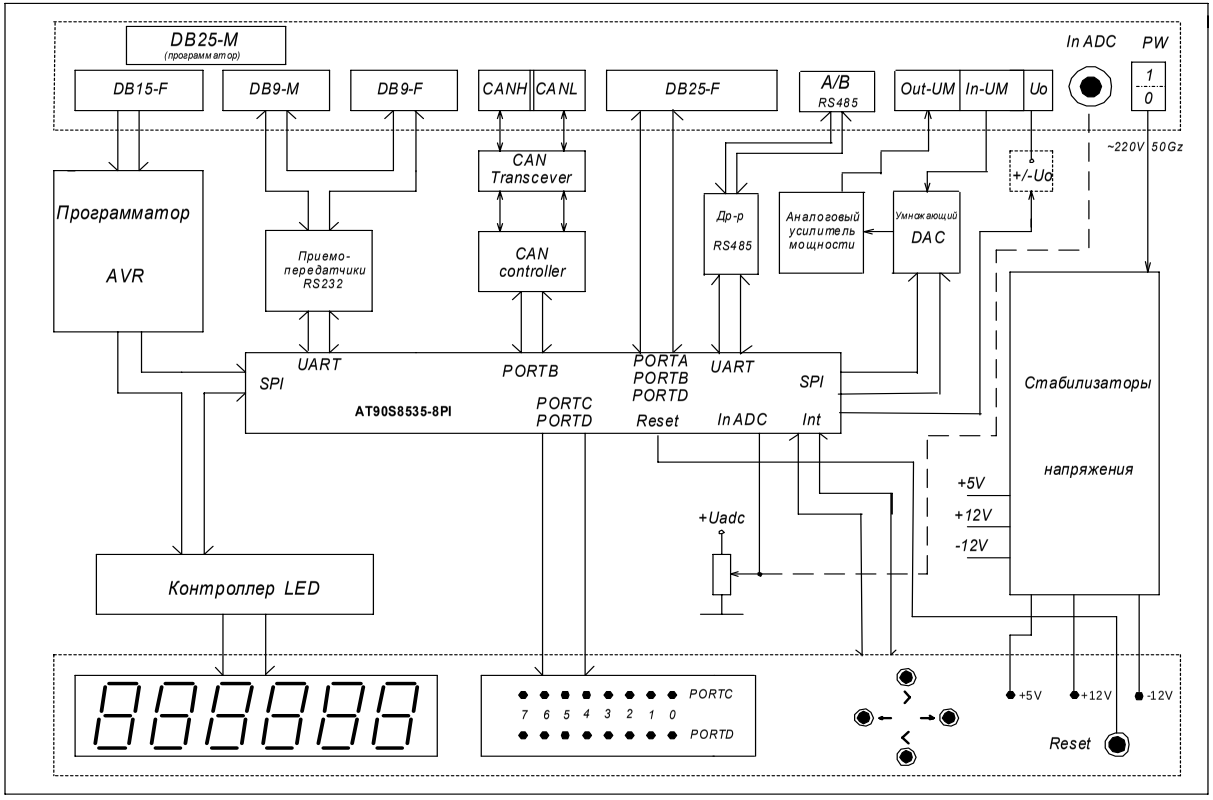

Рис. 1.1. Структурная схема лабораторного стенда

- светодиоды индикации напряжений питания +5, +12, -12;
- управляющая кнопка RESET, предназначенная для организации внешнего системного сброса.

На задней панели лабораторного стенда установлены:

- разъем DB-15M, необходимый для подключения программатора AVR к COM-порту компьютера;
- разъем DB9-P. Разъем предназначен для подключения микроконтроллера через асинхронный приемопередатчик UART к периферийным устройствам: аналогичный лабораторный стенд, модем, принтер и т.п.;
- разъем DB9-M. Разъем предназначен для связи микроконтроллера через приемопередатчик UART с COM- портом компьютера;
- два зажима CANH и CANL, которые обеспечивают подключение стенда к CAN-сети;
- разъем DB25-P. Разъем предназначен для подключения к микроконтроллеру внешних устройств: аналоговых и дискретных датчиков, исполнительных устройств и т.п.;
- зажим U_0, внешний вывод внутреннего источника опорного напряжения $\pm$ 5В;
- зажим $In-UM$, внешний аналоговый вход цифроаналогового преобразователя стенда;
- зажим $Out-UM$, выход аналогового усилителя мощности с коэффициентом усиления $Ku = 3 \cdot K(\beta_i)$, где $K(\beta_i)$ - коэффициент передачи цифроаналогового преобразователя;
- зажимы A/B интерфейса RS-485;
- ручка потенциометра, которая обеспечивает управление напряжением на одном из входов (PA0) аналого–цифрового преобразователя микроконтроллера AVR;
- тумблер PW – выключатель стенда.

Центральным управляющим элементом лабораторного стенда является микроконтроллер AT90S8535. К нему через последовательный интерфейс подключается программатор AVR, выполняющий функции внутрисистемного программирования микроконтроллера. Программатор через разъем DB-15M подключается к COM – порту персонального компьютера.

С помощью последовательного интерфейса SPI микроконтроллера производится передача данных в семисегментный индикатор через LED-контроллер. После передачи интерфейсом SPI необходимого числа байт данных (до 6 байт) необходимо программно формировать сигнал управления индикацией на выходе PB0 микроконтроллера.

Для подключения стенда к персональному компьютеру или подключения к стенду периферийных внешних устройств может использоваться стандарт-

ный интерфейс последовательной передачи данных RS-232, работающий через универсальный приемопередатчик UART микроконтроллера. Интерфейс UART лабораторного стенда поддерживает два режима:

- стенд, подключенный к внешнему устройству с помощью интерфейса RS-232, функционирует как ведомое устройство. В этом режиме он выполняет команды обмена данными, приходящие от внешнего устройства (разъем DB9-M);
- стенд функционирует как ведущее устройство, в этом режиме он управляет обменом данными (разъем DB9-P).

Стенд поддерживает интерфейс CAN–сети. К портам микроконтроллера подключается CAN–контроллер, который через CAN–приемопередатчик осуществляет взаимодействие с другими внешними устройствами по CAN–сети. CAN–сеть представляет собой последовательную асинхронную шину, использующую в качестве среды передачи данных витую пару. При скорости передачи 1 Мбит/с расстояние может достигать 30 м. При меньших скоростях расстояние можно увеличить до километра.

Стенд поддерживает также последовательный интерфейс RS-485. В тех случаях, когда передача данных интерфейсом RS-232 невозможна в силу ряда ограничений таких, как предельная дальность передачи данных (10 метров) и трудность построения многоточечной сети, предполагается применение интерфейса RS-485. Передача данных производится с использованием универсальных приемопередатчиков UART, подключенных через специальный драйвер интерфейса к внешним зажимам A/B. Используя этот интерфейс, данные необходимо передавать по симметричным линиям связи.

Разъем DB25-P (табл. 1.1) позволяет подключать внешние устройства к любым линиям ввода-вывода портов PORTA, PORTB, PORT D микроконтроллера, соответствующим образом программируя функции этих портов. **При выборе этих линий ввода-вывода для передачи сигналов необходимо учитывать альтернативные функции портов микроконтроллера, так как некоторые линии ввода-вывода могут уже использоваться для обмена сигналами.**

Аналоговый усилитель мощности стенда обладает следующими характеристиками: $U_{вых} = 12B$, $P_{вых} = 10Bm$, с его помощью можно непосредственно управлять нагрузкой мощностью до 10 Вт. Сигнал на вход усилителя мощности поступает через умножающий цифроаналоговый

Таблица 1.1

Pin	In/Out
1	PA0
2	PA1
3	PA2
4	PA3
5	PA4
6	PA5
7	PA6
8	PA7
9	PB0
10	PB1
11	PB2
12	PB3
13	PB4
14	PD0
15	PD1
16	PD2
17	PD3
18	PC4
19	PD5
20	PD6
21	PD7
22	PB5
23	PB6
24	PB7
25	GND

преобразователь (ЦАП) стенда. В качестве источника опорного напряжения этого преобразователя можно использовать либо внешнее напряжение, например сигнал обратной связи, либо напряжение от внутреннего стабилизированного источника опорного напряжения. 12–разрядный код ЦАП поступает из микроконтроллера через интерфейс SPI (2 байта данных), для его передачи на управление выходным сигналом необходимо программное формирование сигнала PB1 микроконтроллера. Программное управление полярностью внутреннего источника опорного напряжения производится сигналом PB4 микроконтроллера (логический 0 соответствует положительной полярности). Потенциометр, ручка управления которого находится на задней панели стенда, позволяет регулировать напряжение на входе микроконтроллера PA0 от нуля до опорного напряжения АЦП.

Кнопки управления передней панели формируют сигналы внешних прерываний INT0 (PD2), INT1 (PD3) и сигналы на линиях ввода-вывода PB2, PB3 микроконтроллера (табл. 1.2). Пассивный уровень сигнала – 1, при нажатии какой-либо кнопки низкий уровень сигнала обязательно поступает на один из входов прерываний. Кнопки необходимы для передачи команд, которые управляют выполнением рабочей программы микроконтроллера, изменяя какие-либо параметры, режимы, реализуемые функции и т.п.

Таблица 1.2

Кнопка	INT0 (PD2)	INT1 (PD3)	PB2	PB3
>	0	1	1	1
<	1	0	1	1
→	0	1	0	1
←	1	0	1	0

Таким образом, выполнение стандартных функций, необходимых для реализации типовых задач управления, обеспечивается и микроконтроллером, и дополнительными интерфейсными устройствами лабораторного стенда.

Подготовка программы микроконтроллера на ассемблере для процедуры внутрисистемного программирования может выполняться инструментальным пакетом **AVR Studio.** Пользовательский интерфейс пакета AVR Studio – стандартный для Window's–приложений (рис. 1.2) и содержит полоску меню, панель инструментов и рабочую область. Для активизации функций AVR Studio, необходимых для работы с текстом программы, должны быть выполнены следующие операции:

1. Для работы с текстом программы необходимо создать проект (project). Проект создается с помощью меню **Project** (пункт **New Project**). В появившемся на рабочем поле окне указать тип проекта – AVR Assembler, имя и путь папки, имя проекта. Если проект уже существует, можно открыть его для работы тем же меню **Project** (или меню File). Окно с открытым для работы проектом появится в рабочем поле (рис. 1.3). Кнопкой "Next" необходимо определить опции проекта: Debug Platform – AVR Simulator, тип микроконтроллера Device – AT90S8535.
2. Если в открытом проекте уже есть файлы программ на ассемблере (с расширением .asm), они будут отображены в окне "Workspace" и их можно открыть в рабочем поле двойным щелчком левой кнопки мыши. Создать новые файлы (Create new file) или добавить к проекту созданные ранее файлы (Add existing file) можно с помощью контекстного меню (или меню **Project**), доступ к контекстному меню открывается щелчком правой кнопки мыши в окне Workspace (рис. 1.2). Каждый файл на рабочем поле отображается отдельным окном. Работа с окнами, подготовка и редактирование текстов выполняются стандартными средствами Window's–приложений.

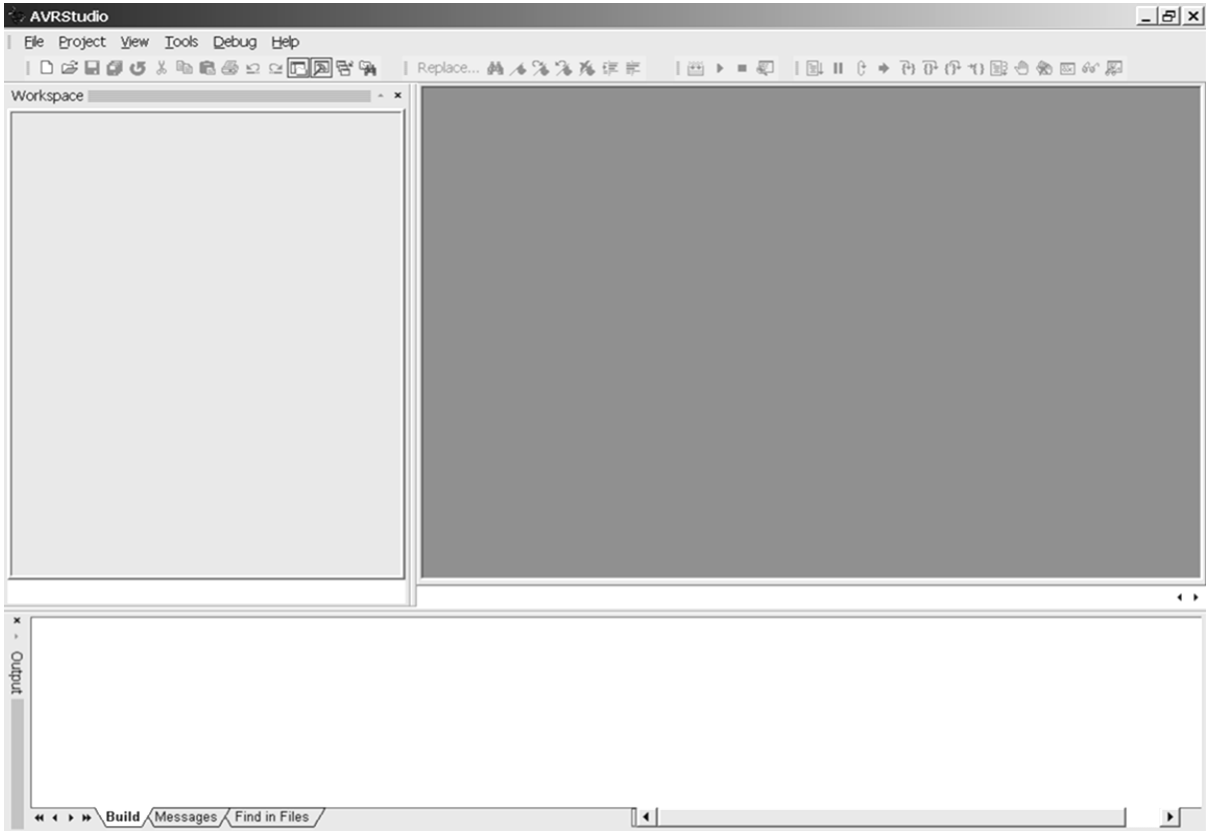

Рис. 1.2. Интерфейс AVR Studio

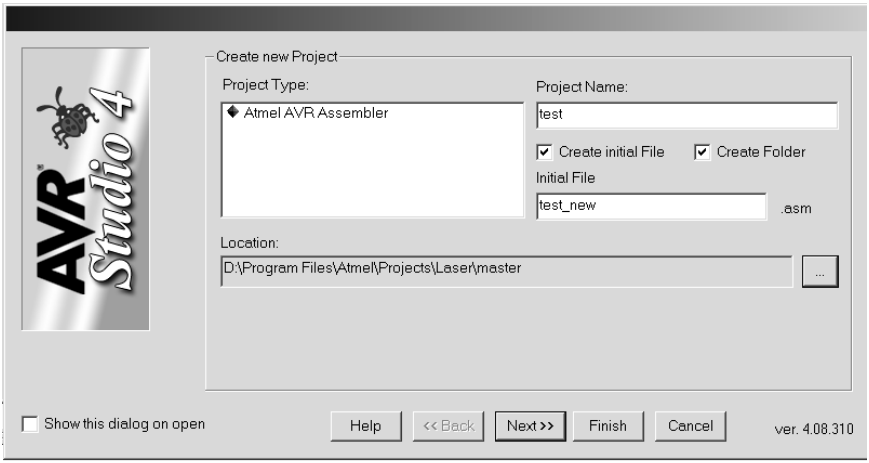

Рис. 1.3. Окно "Create new Project"

3. Компилятор ассемблера может работать только с одним файлом, поэтому необходимо указать для компилятора исходный файл в контекстном меню (Set as Entry File). После этого можно выполнить запуск компилятора из меню **Project** – пункт **Build (F7)** или соответствующей кнопкой панели инструментов. По завершении ассемблирования компилятор сформирует в окне "Output" (рис. 1.2) сообщение о результатах. Если синтаксических ошибок нет, выходной файл сформирован. Обнаруженные ошибки будут указаны компилятором (окно "Output"), в сообщении указываются тип ошибки и номер строки в исходной программе. Отметить строку с ошибкой в исходном тексте можно двойным щелчком левой кнопки мыши на сообщении об ошибке.

Предварительная проверка подготовленной программы после устранения синтаксических ошибок производится с помощью симулятора (Debug Platform – AVR Simulator). Запуск симулятора производится из меню **Project** – пункт **Build and Run (Ctrl+F7)** или соответствующей кнопкой панели инструментов. Включение различных функций симулятора, режимы их моделирования определяются различными меню пакета AVR Studio или кнопками панели инструментов (более подробную информацию об этом можно найти в меню Help).

Отображение результатов работы программы в симуляторе производится в окне "Workspace" переключением в режим I/O (рис. 1.4). В этом окне можно отображать содержимое всех регистров и ячеек памяти микро-

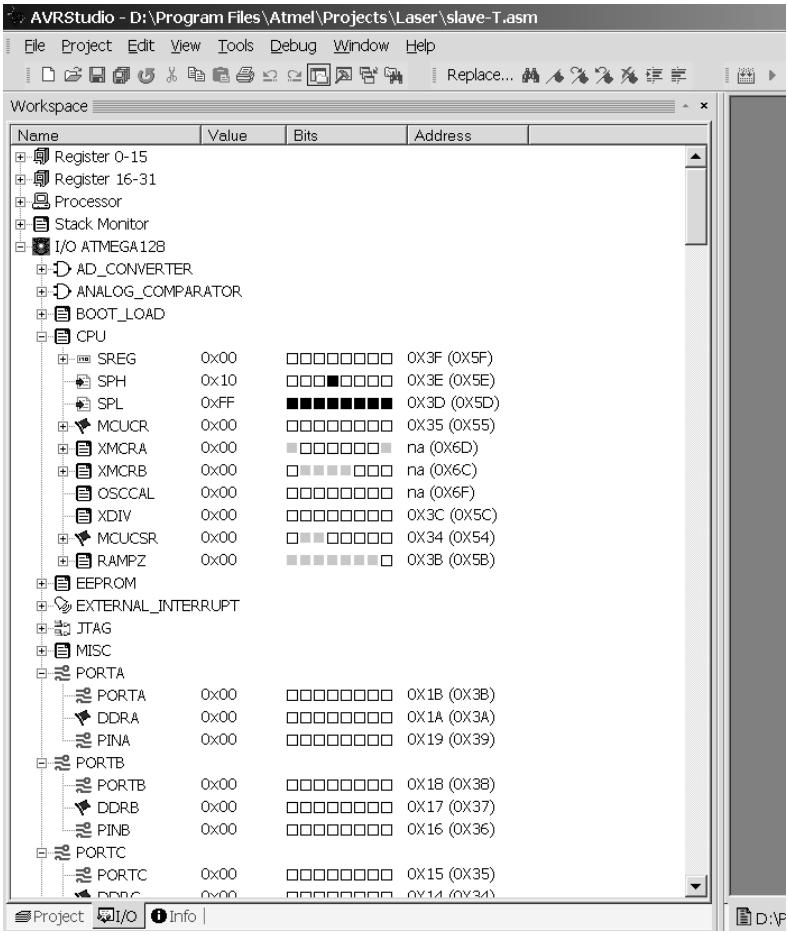

Рис. 1.4. Окно I/O

контроллера. При использовании пошагового режима выполнения программ на любом этапе можно производить изменение данных вводом с клавиатуры PC новых значений в двоичном, шестнадцатеричном или десятичном формате. Аналогичные операции можно выполнять с помощью окон "Watch", формируя в этих окнах произвольный и более удобный для работы с программой набор отображаемых элементов микроконтроллера.

После проверки корректности программы симулятором можно с помощью программатора произвести запись подготовленного Hex–файла в память

микроконтроллера. Необходимо учитывать, что симулятор не позволяет моделировать все необходимые функции. Окончательное тестирование программы может проводиться только на лабораторном стенде с микроконтроллером.

Для записи подготовленного Hex–файла в микроконтроллер необходимо подключить лабораторный стенд к COM-порту компьютера (в стенде используется разъем DB15F с надписью AVR studio Prog), подключить стенд к сети 220В и включить питание. После включения стенда программатор AVR Studio может работать, его запуск производится из меню **Tools** пунктом **AVR Prog**. Управление программатором производится в открывшемся после запуска окне **AVRprog** на рабочем поле.

В окне программатора необходимо выбрать записываемый в микроконтроллер Hex–файл (кнопка Browse, файл после компиляции находится в папке проекта, с которым Вы работаете). В строке **Device** выбрать тип микроконтроллера (AT90S8515) и запустить программатор в режиме программирования флэш-памяти программ кнопкой **Program** в области **Flash** окна программатора. Для управления программированием ППЗУ в окне программатора предусмотрена область **EEPROM.** Устройство начинает работать сразу после завершения программирования.

2. ЛАБОРАТОРНАЯ РАБОТА № 1 ВВОД-ВЫВОД ДАННЫХ В ПАРАЛЛЕЛЬНОМ ФОРМАТЕ

2.1. Цель работы

В работе изучаются особенности управления параллельными портами ввода-вывода микроконтроллеров AVR, использование сигналов внешних прерываний для управления процедурами ввода-вывода данных в параллельном формате.

2.2. Параллельные порты ввода-вывода микроконтроллера AT90S8535

Микроконтроллер содержит 32 линии ввода-вывода, объединенные в 4 двунаправленных порта (A, B, C, D) [1]. Управление каждым портом производится тремя регистрами порта из файла регистров ввода-вывода, символические имена этих регистров содержат наименование порта. Функции регистров приведены ниже для обобщенного порта с именем X, где X – символическое имя порта: A, B, C или D.

Регистр управления с символическим именем DDRX программно доступен и для чтения, и для записи, DDRX определяет направление передачи данных: 0 – ввод, 1 – вывод. Каждый бит DDRX (DDX0 – DDX7) управляет соответствующей линией ввода-вывода и программируется независимо, т.е. значение каждого бита и направление передачи сигналов может задаваться произвольно, начальное значение всех битов DDRX – нулевое (ввод данных). В процессе работы чтением содержимого регистра DDRX можно определить направление передачи данных по соответствующим линиям ввода-вывода в данный момент времени.

Регистр ввода данных PINX программно доступен только для чтения и обеспечивает считывание сигналов, поступающих в данный момент времени на соответствующие линии ввода-вывода (например, линия PA2 в режиме ввода формирует бит PINA2 регистра ввода данных). Хранения данных регистр PINX не выполняет, передавая при чтении текущие состояния сигналов.

Регистр вывода данных PORTX программно доступен и для чтения, и для записи, обеспечивает хранение данных и выдачу их в режиме вывода на соответствующие линии ввода-вывода (например, PORTA4 – PA4). При чтении PORTX передает данные, ранее записанные в этот регистр для вывода.

Пример настройки портов микроконтроллера:

```
    ldi r16, 0b11101000 ;запись двоичной константы в r16
    out DDRA, r16       ;пересылка содержимого r16 в регистр управления
                        ;порта A
    clr r16         ;очистка регистра  r16
    out DDRB, r16       ;все линии порта B на ввод данных
    ldi r16, 0xf0       ;запись шестнадцатеричной константы в r16
    out DDRC, r16       ; 4 младших бита порта C на ввод,
                        ; 4 старших бита порта C на вывод
    ser r16             ; все биты регистра r16 в 1
    out DDRD, r16       ; порт D на вывод
lb1:  out PORTA, r16
lb2:  in  r16, PINA
```

Команда с меткой "lb1:" передает в выходной регистр (PORTA) порта A из регистра общего назначения r16 константу 0xff, которая определит единичный уровень сигнала на линиях PA3, PA5-PA7, настроенных на вывод данных. В остальных линиях PA0-PA2, PA4, настроенных на ввод данных, запись единиц в соответствующие биты выходного регистра PORTA подключит внутренние резисторы порта, которые определяют "пассивный высокий" уровень сигнала. При отсутствии внешнего источника сигнала на этих линиях задается высокий уровень внутренними резисторами порта. Внешний источник сигнала выходным током, превышающим единицы микроампер, может сформировать сигнал как низкого уровня, так и высокого уровня.

Команда с меткой "lb2:" пересылает в регистр r16, входные сигналы порта из регистра ввода PINA. При чтении PINA в битах, которые настроены на вывод данных (PINA3, PINA5-PINA7), передается содержимое выходного регистра PORTA; в битах, настроенных на ввод данных (PINA0-PINA2, PINA4), передается состояние входных сигналов.

Таким образом, функциональная гибкость портов обеспечивается независимым управлением каждой из 32 линий ввода-вывода микроконтроллера, допускается в процессе работы изменять направление передачи сигналов изменением соответствующих данных в регистрах портов. При выборе режимов ввода-вывода сигналов необходимо учитывать альтернативные функции портов микроконтроллера. Определенные линии ввода-вывода аппаратно связаны с интерфейсами устройств микроконтроллера: АЦП, таймеров, последовательных приемопередатчиков и т.д. При использовании этих устройств соответствующие линии ввода-вывода жестко закреплены за их интерфейсами и не могут выполнять какие-либо другие функции обмена данными.

Достаточно часто необходимость в обработке сигналов ввода-вывода зависит от изменения каких-либо входных сигналов, связанных, например, с

нажатием кнопки управления, срабатыванием аварийного датчика и т.п. Для того чтобы не производить постоянную обработку этих сигналов, можно использовать процедуры вызова внешних прерываний [1]. Микроконтроллер лабораторного стенда позволяет реализовать два внешних прерывания: INT0 (PD2) и INT1 (PD3). Параметры сигналов PD2 и PD3, формирующие флаги внешних прерываний, определяются парами битов ISC01, ISC00 (биты 1 и 0) и ISC11, ISC10 (биты 3 и 2) регистра управления MCUCR:

00 – низкий уровень сигнала устанавливает флаг вызова внешнего прерывания,

01 – не используется,

10 – падающий фронт сигнала устанавливает флаг вызова внешнего прерывания,

11 – нарастающий фронт сигнала устанавливает флаг вызова внешнего прерывания.

В таблице векторов прерываний по адресам $001 (INT0), $002 (INT1) указываются метки подпрограмм обработки прерываний. Кроме этого, разрешение обработки сигналов внешних прерываний определяется битами 6 (INT0) и 7 (INT1) в регистре управления GIMSK и флагом глобального разрешения прерываний I регистра SREG. Соответствующие битам регистра MCUCR изменения сигналов внешних прерываний формируют флаг вызова подпрограмм обработки. Если прерывание разрешено в регистрах GIMSK и SREG, работа текущей программы в микроконтроллере приостанавливается, и управление передается подпрограмме обработки, адрес которой указан в таблице векторов прерываний.

Пример программы обслуживания внешних прерываний:

```
.ORG $000
    rjmp   reset
    rjmp   INT0
    rjmp   INT1
.ORG $011
reset:  ldi r16, low(RAMEND)
        out     SPL, r16
        ldi r16, high(RAMEND)
        out SPH, r16      ; определить в указателе стэка адрес RAMEND
        ser  r16
        out PORTD, r16 ; пассивный высокий в порте D
    clr r16
    out DDRA, r16       ; порт A на ввод
    out DDRD, r16       ; порт D на ввод
    out     MCUCR, r16      ; внешние прерывания по низкому уровню
```

```
    ldi r16, 0b11000000
    out GIMSK, r16        ;разрешение вызова INT1, INT0

    .  .  .  .  .  .  .
    sei                   ;флаг глобального разрешения прерываний
    .  .  .  .  .  .  .

 ;п/программа вектора прерывания INT0
INT0: push   r16
    in  r16, PINA      ;пересылка входных сигналов порта A в r16 для обработ-
ки                     ;по прерыванию INT0
    .  .  .  .  .  .  .
    pop    r16
    reti               ;завершение прерывания

; п/программа обработки вектора прерывания INT1
INT1:
    .  .  .  .  .  .  .
    reti
```

В примере вектор прерывания "reset" передает управление программе инициализации при начальном сбросе. Программа инициализации (метка "reset") производит следующие операции:

- загрузку регистра указателя стека (SPH, SPL), необходимого для выполнения вызовов любых подпрограмм;
- настройку портов A и D на ввод данных;
- загрузку регистра MCUCR для формирования флагов вызова векторов внешних прерываний по низкому уровню сигналов, поступающих на входы PD2, PD3;
- загрузку регистра GIMSK битами разрешения вызова векторов внешних прерываний.

При поступлении сигнала низкого уровня на PD2, настроенного на ввод данных, произойдет автоматический вызов подпрограммы вектора прерывания по метке " INT0:". Эта подпрограмма сохраняет в стеке содержимое регистра r16, которое может измениться, считывает сигналы порта A в регистр r16. Содержимое этого регистра должно анализироваться и формировать необходимую реакцию микроконтроллера, перед завершением подпрограммы производится восстановление содержимого r16 из стека.

Другой важный фактор, который необходимо учитывать при организации обмена данными, связан с временными характеристиками формируемых сигналов. Обработка сигналов и их прием или выдача должны производиться строго периодически со временем, определяемым интервалом дискретиза-

ции, инерционностью исполнительных устройств, требуемым спектром сигналов и т.д. Время реализации алгоритма обработки данных процессором для определения периодичности обмена в большинстве случаев использовать затруднительно, так как это время может изменяться в значительных пределах. Более эффективно периодичность обмена данными задавать таймерами, которые и предназначены для решения таких задач.

2.3. Формирование временных интервалов таймером

В микроконтроллере лабораторного стенда имеется 3 таймера, каждый таймер управляется независимо, может программироваться в различные режимы работы и использоваться для решения разнообразных задач. В качестве примера рассмотрим применение таймера 2 для формирования периодичности вывода данных параллельного формата в порт С. Управление работой таймера 2 так же, как и других элементов микроконтроллера, производится управляющими регистрами, находящимися в файле регистров ввода-вывода.

Определим условия задачи следующим образом: производить выдачу байта данных в параллельном формате через порт С периодически с интервалом времени 20 мс, подготовленный для вывода байт данных находится в регистре r20. Управление выводом целесообразно производить вектором прерывания, вызов которого будет формировать таймер 2 через заданный интервал времени (20 мс), обеспечивая необходимую периодичность выдачи данных.

Периодический вызов вектора прерывания достаточно просто реализуется при использовании вектора канала сравнения TIM2_COMP. Таймер 2 в этом случае работает в режиме суммирующего счетчика импульсов, поступающих на его вход. Когда число входных импульсов становится равным коду, записанному в регистр OCR2 канала сравнения, формируется флаг вызова вектора прерывания и при установленном разрешении в бите CTC2 (бит 3 регистра TCCR2) производится сброс таймера в исходное состояние. Разрешение на обработку этого прерывания должно быть также установлено битом разрешения TOIE2 (бит 6 регистра TIMSK) и флагом глобального разрешения прерываний I.

Периодичность вызова вектора прерывания T_{comp2} зависит от частоты входного сигнала таймера и кода канала сравнения OCR2. Параметры входного сигнала таймера определяются битами CS22, CS21, CS20 (биты 2, 1, 0 регистра TCCR2). При коде 000 – таймер остановлен, при коде 111 – на вход таймера поступают импульсы частотой $f_{clk}/1024$, где f_{clk}=7,3728 МГц – частота тактового генератора микроконтроллера. Для получения необходимого периода T_{comp2} (мкс) код OCR2 можно определить следующим образом:

$$OCR2 = \frac{T_{comp2} \times f_{clk}}{1024} = \frac{20000 \times 7{,}3728}{1024} = 144 \rightarrow (0x90).$$

Таким образом, для периода 20 мс необходимо записать в регистр OCR2 число 0x90 и установить бит разрешения прерывания TOIE2 в регистре TIMSK. После запуска таймера загрузкой регистра TCCR2 двоичным кодом 0b00001111 каждые 20 мс будет производиться вызов вектора прерывания COMP2 и сброс таймера в исходное состояние для отсчета следующего периода.

```
.ORG $000
    rjmp    reset
.ORG $003
    rjmp    COMP2       ;вектор прерывания по каналу сравнения таймера 2
.ORG $011
reset:  ldi r16, low(RAMEND)
    out     SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16      ; определить в указателе стэка адрес RAMEND
    ser  r16
    out DDRC, r16       ; порт C на вывод
    ldi r16, 0x90
    out OCR2, r16       ; макс. значение состояния счетчика таймера 2
    ldi r16, 0b01000000
    out TIMSK, r16      ; разрешение вызова вектора прерывания COMP2
    ldi r16, 0b00001111
    out TCCR2, r16      ; запуск таймера 2 (начало отсчета 20 мс)
    . . . . . .
    sei             ;флаг глобального разрешения прерываний
    . . . . . .
COMP2:
    out PORTC, r20      ;
    clr r20         ;
    reti            ;
```

2.4. Порядок выполнения лабораторной работы

В соответствии с вариантом индивидуального задания, выданного преподавателем, подготовить программу вывода данных в параллельном формате в порт C микроконтроллера с возможностью управления работой кнопками передней панели лабораторного стенда (пуск, останов процедуры изменения выходных данных, управление параметрами вывода и т.п.). Функции кнопок

управления можно задавать произвольно. Целесообразно использовать распределение функций управления аналогичное функциям в пультах дистанционного управления различных радиоэлектронных устройств: кнопки по вертикали – выбор изменяемого параметра, кнопки по горизонтали – увеличение/уменьшение выбранного параметра или пуск/останов процедуры.

Алгоритмы формирования данных, передаваемых в порт С: "бегущий огонь" (слева-направо, справа-налево с разным числом включенных светодиодов и/или разной скоростью переключения), увеличение/уменьшение числа включенных светодиодов, увеличение/уменьшение двоичного кода выходного байта по заданному алгоритму с заданной скоростью переключения. Обработка сигналов кнопок управления должна производиться с использованием процедуры обработки внешних прерываний и алгоритма "антидребезга".

3. ЛАБОРАТОРНАЯ РАБОТА № 2
ВВОД-ВЫВОД ДАННЫХ В ПОСЛЕДОВАТЕЛЬНОМ ФОРМАТЕ

3.1. Цель работы

В работе изучаются алгоритмы работы последовательных интерфейсов ввода-вывода SPI, UART микроконтроллеров AVR и управление этими интерфейсами ввода-вывода данных.

3.2. Синхронный последовательный интерфейс SPI

Последовательный интерфейс SPI [1] обеспечивает синхронный ввод-вывод данных в последовательном формате через линии ввода-вывода порта B сигналами PB7 (SCK), PB5 (MOSI), PB6 (MISO), PB4 (SS). Контроллер при обмене данными может работать в режиме "ведущий" (master) или "ведомый" (slave). Структурная схема, поясняющая алгоритмы работы интерфейса, приведена на рис. 3.1.

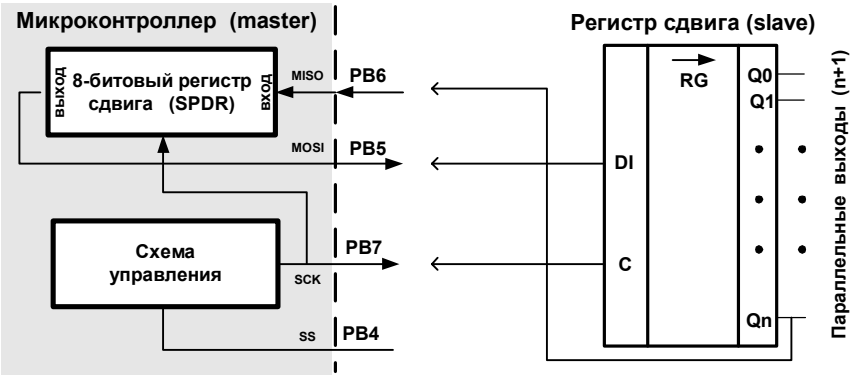

Рис. 3.1. Интерфейс SPI

SPI-master управляет обменом данных. В лабораторном стенде эти функции всегда выполняет микроконтроллер, SPI-slave – приемные сдвиговые регистры цифрового индикатора и ЦАП, которые дополнительно требуют передачи сигнала управления "load" (PB0 для индикатора, PB1 для ЦАП) после окончания передачи сообщения в последовательном формате. SPI-master при работе интерфейса:

- на выходе SCK формирует стробирующую обмен данными последовательность тактовых импульсов;
- на выход MOSI передает информационную 8-битовую последовательность из регистра SPDR;
- с входа MISO принимает в регистр SPDR 8-битовую последовательность.

Для SPI-master в PB5 должен быть задан режим вывода, в PB6 – режим ввода. Если PB4 конфигурируется как выход (DDB4=1), сигнал SS не используется и PB4 может работать в стандартном режиме вывода. В режиме ввода PB4 может использоваться для переключения микроконтроллера в режим SPI-slave. Приемные сдвиговые регистры (SPI-slave) постоянно находятся в режиме приема данных по сигналам SCK, а формирование сигнала управления "load" обеспечивает запись данных только в то устройство, которому они адресованы.

Управление параметрами и режимами интерфейса производится регистром SPCR. Младшие биты SPR1 (бит 1), SPR0 (бит 0) определяют тактовую частоту SCK, при значении этих битов 10 частота $f_{SCK}= f_{CLK}/16$. Биты CPOL (бит 3), CPHA (бит 2) определяют рабочие параметры сигнала SCK, для корректной работы приемных регистров необходимые значения 11. Старшие 4 бита регистра управления SPCR должны программироваться значениями 1101. Таким образом, рекомендуемое для лабораторного стенда значение байта управления регистра SPCR – 0b11011110. Если управление работой SPI не предусматривает использование вектора SPI_STC, бит 7 байта управления должен быть нулевым, при этом вызов вектора SPI_STC будет запрещен. Запись байта данных в регистр SPDR программой микроконтроллера автоматически запускает процесс передачи этого байта интерфейсом SPI.

Пример: передать в последовательном формате из микроконтроллера во внешний регистр (ЦАП) 2 байта данных, находящихся в ОЗУ по адресам: $060, $061 с формированием сигнала завершения цикла передачи "load".

Задачу передачи данных в последовательном формате можно решить использованием интерфейса микроконтроллера SPI, для адресации данных в ОЗУ целесообразно использовать один из регистров косвенной адресации, например, регистр X.

```
.NOLIST
.INCLUDE "8535def.inc"
.LIST
.DEF   temp = r16          ;определить символическое имя temp регистру r16
.CSEG
.ORG $000
    rjmp    init           ;прерывание по reset
.ORG $00a
    rjmp    spi_stc        ;прерывание по завершению передачи байта
.ORG $011
init:    ldi  temp, low(RAMEND)
    out   SPL, temp
    ldi  temp, high(RAMEND)
    out      SPH, temp          ; определить в указателе стека адрес RAMEND
    clr temp
    out      PORTB, temp          ; PORTB ≡ 0
    ldi  temp, $ff
    out      DDRB, temp       ; PORTB на вывод данных
    ldi   temp, 0b11011110
    out  SPCR, temp       ; инициализация SPI
    sei                ; установить флаг общего разрешения прерываний
; бессодержательный бесконечный цикл, может быть заменен любой про-
граммой
main:  nop
    rjmp    main
.  .  .  .  .  .  .
;запуск цикла с выводом первого байта в последовательном формате
start:   clr  XH
    ldi  XL, $60         ; занести начальный адрес данных в регистр X
    ld   temp, X+        ; занести в temp байт с постинкрементом адреса в X
    out  SPDR, temp       ; переслать байт в регистр данных SPI
    ret

;п/программа для вектора прерывания SPI
spi_stc:    cpi  XL, $62
        breq    end          ; перейти, если цикл завершен
        ld   temp, X+  ; занести в temp байт с постинкрементом адреса в X
        out  SPDR, temp       ; переслать байт в регистр данных SPI
        reti           ; возврат из прерывания
```

```
; завершение цикла
end:    sbi PORTB, PB1      ; установить сигнал load
    nop
    cbi PORTB, PB1      ; сбросить сигнал load
    reti                ; возврат из прерывания
```

Как и в приведенных ранее примерах, программа содержит файл определения символических имен "8535def.inc", далее директивой компилятора определено символическое имя "temp" для регистра r16. Первые операторы программы – используемые векторы прерываний. По метке "init:" производится настройка используемых средств микроконтроллера, бессодержательный цикл по метке "main:" показывает, что процессор может выполнять любые программы одновременно с выводом данных по SPI. Запуск интерфейса SPI производится программно вызовом подпрограммы по метке "start:". Как только байт данных будет записан в регистр SPDR, начнется формирование сигналов интерфейса SPI без участия процессора. После завершения вывода байта данных произойдет вызов вектора прерывания "spi_stc:", который запишет новый байт данных в регистр SPDR. Завершение вывода второго байта и повторный вызов вектора прерывания передает управление программе завершения цикла (метка "end:") и формирования сигнала "load". После этого интерфейс готов к запуску следующего цикла вывода данных.

Асинхронный последовательный интерфейс UART

Интерфейс UART [1] обеспечивает полнодуплексную передачу данных через линии ввода-вывода порта D сигналами: вход приемника – RXD (PD0) и выход передатчика – TXD (PD1). Приемник и передатчик интерфейса работают независимо друг от друга в асинхронном режиме со стандартными скоростями передачи данных и стандартным форматом сообщения, включающим стартовый признак, от 5 до 8 информационных бит, стоповый признак. Выбранная скорость передачи данных должна быть указана в параметрах настройки интерфейса UART. Более подробная информация о работе интерфейса может быть найдена в описании любого асинхронного приемопередатчика, например, К580ВВ51 или интерфейса RS-232C.

Регистр данных интерфейса (регистр UDR) содержит два независимых регистра: при записи в UDR данные поступают в регистр передачи интерфейса, при чтении считывается содержимое другого регистра, обеспечивающего прием данных интерфейсом.

Запись байта данных в регистр UDR автоматически начинает цикл передачи этого байта: стартовый бит – биты данных – стоповый бит. После завершения передачи и готовности к передаче следующего байта данных формируется флаг TXC (бит 6 регистра USR). Контролировать флаг завершения передачи TXC можно программно чтением регистра USR либо вектором

прерывания UART_TXC. Для вызова вектора прерывания должен быть установлен бит разрешения прерывания TXCIE (бит 6 регистра UCR) и флаг глобального разрешения прерывания I.

Прием байта данных и контроль формата сообщения производит приемник интерфейса UART. После завершения приема байт данных помещается в выходной регистр с тем же символическим именем UDR и формируется флаг завершения приема RXC (бит 7 регистра USR). Этот флаг также можно контролировать программно или вектором прерывания UART_RXC. Для вызова вектора прерывания должен быть установлен бит разрешения прерывания RXCIE (бит 7 регистра UCR) и флаг глобального разрешения прерывания I. Если байт данных не будет считан из UDR до завершения обработки следующего сообщения приемником, байт данных будет заменен следующим принятым байтом.

Таким образом, наиболее эффективное использование интерфейса UART может обеспечиваться векторами прерываний: вектор завершения передачи должен записывать подготовленный байт данных в регистр UDR для передачи, а вектор завершения приема – принимать из регистра с этим же символическим именем UDR байт данных из приемника и производить его обработку. Для настройки интерфейса UART в этом случае необходима запись байта управления 0xd8 в регистр UCR. Скорость обмена данными определяется тактовой частотой микроконтроллера (7,3728 МГц) и значением константы в регистре UBRR; для скорости 115200 бит/с в регистр UBRR необходимо записать константу 03.

3.4. Порядок выполнения лабораторной работы

В соответствии с вариантом индивидуального задания подготовить программу вывода данных в последовательном формате через интерфейсы SPI и UART микроконтроллера. Для формирования выводимых данных использовать программу преобразования данных лабораторной работы № 1 со всеми необходимыми функциями кнопок управления.

В качестве приемника интерфейса SPI использовать регистр цифрового шестиразрядного светодиодного индикатора. Для корректного отображения необходимо каждый байт данных перед выводом преобразовать в двоично-десятичный формат (три десятичных разряда), полученный код десятичной цифры преобразовать в код управления семисегментным индикатором. Коды управления семисегментными индикаторами в шестнадцатеричном формате приведены в табл. 3.1.

Таблица 3.1

Цифра	0	1	2	3	4	5	6	7	8	9
Код	0x3f	0x06	0x5b	0x4f	0x66	0x6d	0x7d	0x07	0x7f	0x6f

При подготовке процедуры вывода данных на светодиодный индикатор необходимо учитывать инерционность зрения. Частота обновления данных на индикаторе должна быть не более 3–4 Гц. Если алгоритм формирования предполагает слишком высокую скорость преобразования данных, необходимо модифицировать используемую из лабораторной работы № 1 программу и снизить частоту до приемлемой для восприятия. Необходимые изменения можно внести программной настройкой таймеров и соответствующим увеличением временных интервалов. Если параметры таймеров не позволяют увеличить требуемые временные интервалы, возможно использование дополнительной программно реализуемой задержки: запуск процедуры изменения данных не первым прерыванием таймера, а прерыванием с номером N. Это позволит в N раз увеличить длительность временных интервалов.

Для интерфейса UART подготовить программу обмена данными с компьютером через COM-порт и интерфейсное Window's-приложение Hyper Terminal по следующему алгоритму: прием заданного числа байтов и пересылка в обратном направлении после обработки. Данные для вывода необходимо формировать программным модулем лабораторной работы № 1.

4. ЛАБОРАТОРНАЯ РАБОТА № 3
ВВОД-ВЫВОД АНАЛОГОВЫХ СИГНАЛОВ

4.1. Цель работы

Изучение вопросов построения интерфейсов для приема аналоговых сигналов встроенным АЦП микроконтроллера и для формирования выходных аналоговых сигналов встроенными модуляторами ШИМ и внешним ЦАП.

4.2. Аналого-цифровой преобразователь микроконтроллера

АЦП микроконтроллера разрядностью 10 бит работает по алгоритму последовательных приближений [1], погрешность преобразования – не более 2 единиц младшего значащего разряда, время преобразования 65 мкс – 260 мкс. АЦП совместно со встроенным аналоговым мультиплексором обеспечивает преобразование в 10–разрядный двоичный код сигналов по одному из 8 аналоговых входов (порт A) в диапазоне напряжений от 0 (AGND) до опорного (AREF).

Опорное напряжение должно лежать в диапазоне от 2 В до напряжения питания AVCC (5 В). Код АЦП $N_{АЦП}$=0x000 соответствует нулевому входному сигналу $U_{АЦП}$, максимальный код 0x3FF соответствует сигналу, равному опорному AREF минус вес единицы младшего значащего разряда:

$$U_{АЦП}= (N_{АЦП}*AREF)/2^{10}.$$

Выходной код $N_{АЦП}$ хранится в двухбайтовом регистре ADC (ADCL – младшие 8 бит результата, ADCH – старшие 2 бита результата). Чтение данных из регистра результата ADC, как и других двухбайтовых регистров, должно начинаться обязательно с младшего байта.

Номер входа мультиплексора, с которого поступает сигнал для преобразования в АЦП, определяется тремя младшими битами MUX2, MUX1, MUX0 управляющего регистра ADMUX. Любой из восьми входов может быть выбран через ADMUX записью в него соответствующего кода в любой момент времени, однако переключение входов фактически производится только после завершения очередного цикла преобразования АЦП.

АЦП может работать в режиме однократного преобразования или циклически с автоматическим повторным запуском после каждого преобразования. Управление работой АЦП производится через управляющий регистр ADCSR, биты которого определяют параметры и режимы работы. Время преобразования (тактовая частота АЦП) зависит от битов ADPS2, ADPS1, ADPS0 (биты 2, 1, 0), рекомендуемое значение – 110. Так же, как и при ис-

пользовании других аппаратных средств микроконтроллера, наиболее эффективно управление работой АЦП через вектор завершения преобразования – ADC. Этот вектор прерывания должен предусматривать чтение из регистров ввода-вывода ADCL, ADCH результатов преобразования, переключение мультиплексора на другой вход порта А изменением младших битов регистра ADMUX (при необходимости), запуск следующего цикла преобразования АЦП, обработку полученных результатов преобразования. В этом случае целесообразно устанавливать режим однократного преобразования ADFR=0 (бит 5 регистра ADCSR), а запуск преобразования производить установкой бита ADSC (бит 6 регистра ADCSR). Первый запуск АЦП должен производиться из программы инициализации, рекомендуемое значение байта управления АЦП – ADCSR≡0b11011110.

Величина опорного напряжения AREF в лабораторном стенде равна +5 В, для тестирования АЦП можно использовать потенциометр, позволяющий регулировать напряжение на входе PA0 в диапазоне 0-5 В.

4.3. Цифроаналоговый преобразователь лабораторного стенда

Цифроаналоговый преобразователь не входит в состав аппаратных средств микроконтроллера, в лабораторном стенде внешний 12–разрядный умножающий цифроаналоговый преобразователь (ЦАП) подключен к микроконтроллеру через интерфейс SPI. После приема из SPI двухбайтового кода, для передачи его на управление выходным сигналом необходимо программное формирование сигнала PB1 микроконтроллера. Программное управление полярностью внутреннего источника опорного напряжения ЦАП производится сигналом PB4 микроконтроллера (логический 0 соответствует положительной полярности). Выходной сигнал ЦАП через усилитель мощности с K_U=3 и $P_{ВЫХ}$ до 10 Вт поступает на выходной зажим задней панели лабораторного стенда. Возможны два режима применения ЦАП с усилителем мощности.

Если на внешний аналоговый вход ЦАП подать опорное напряжение стенда U_0 , на выход усилителя мощности будет поступать сигнал $U_{ВЫХ}$, параметры которого будут зависеть только от кода управления $N_{ЦАП}$:

$$U_{ВЫХ} = 3*(N_{ЦАП}* U_0)/2^{12}.$$

В этом режиме характер изменения аналогового выходного сигнала определяется только алгоритмом формирования кода $N_{ЦАП}$.

Если на аналоговый вход ЦАП подать внешний входной сигнал, то для этого сигнала ЦАП с усилителем мощности будет выполнять функции усилителя с программируемым коэффициентом усиления:

$$U_{ВЫХ} = 3*(N_{ЦАП}* U_{ВХ})/2^{12}, \quad K(N_{ЦАП}) = (3*N_{ЦАП})/2^{12}.$$

В этом режиме код $N_{ЦАП}$ будет определять коэффициент передачи $K(N_{ЦАП})$.

Таким образом, для применения ЦАП с усилителем мощности необходимо сформировать код $N_{ЦАП}$, через интерфейс SPI передать этот код в регистр ЦАП и после формирования сигнала параллельной выдачи в ЦАП (PB1) этот код будет определять параметры аналогового выходного сигнала усилителя мощности.

4.4. Выходной сигнал ШИМ

Применение аналоговых усилителей для формирования сигналов управления с большей выходной мощностью ограничено низким кпд. Как известно, при токах единицы – десятки ампер необходимо применение не аналоговых, а ключевых выходных каскадов, обладающих кпд более 90%. В этих случаях для формирования импульсных выходных сигналов, среднее значение ($U_{ср}$) которых может быть представлено непрерывной функцией управляющих сигналов, применяют широтно-импульсную модуляцию (ШИМ):
где T – период ШИМ,

$$U_{ср} = \frac{t_u}{T} * E,$$

t_u – длительность выходного импульса ШИМ,

E – напряжение питания выходного ключевого каскада (рис. 4.1).

При использовании ШИМ необходимо учитывать, что формируемый выходной сигнал импульсный, и в спектре выходного сигнала появляются дополнительные спектральные составляющие с частотами, кратными частоте ШИМ ($f=1/T$).

Один из режимов работы таймеров 1, 2 микроконтроллера предназначен для реализации ШИМ, это позволяет подготовленными кодами управления формировать выходные сигналы с ШИМ. Таймер 1 может реализовать два канала ШИМ с независимыми кодами управления (каналы сравнения A и B), таймер2 – один канал ШИМ.

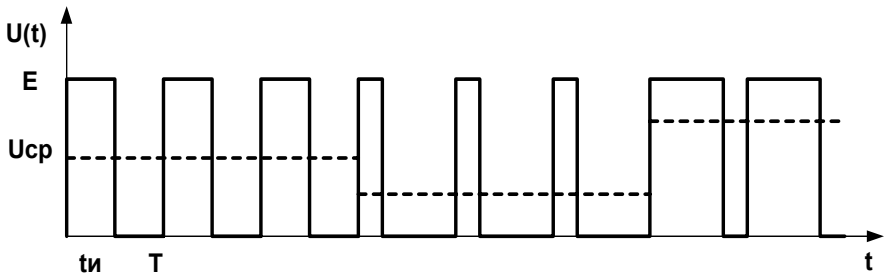

Рис. 4.1. Выходной сигнал ШИМ

Рассмотрим применение таймера 2 в режиме модулятора ШИМ. Код управления *N*, определяющий среднее значение выходного напряжения, должен записываться в регистр канала сравнения OCR2:

$$Ucp = E\frac{N}{255}.$$

Таким образом, изменяя код управления *N* от 0 до 255, будем формировать среднее значение выходного напряжения в диапазоне от 0 до *E* с дискретностью 1/255. Выходные импульсы ШИМ будут поступать на выход OC2 (PD7) микроконтроллера. Частота выходных импульсов (*f=1/T*) будет зависеть от частоты входных импульсов таймера 2. При значении битов управления CS22, CS21, CS20 регистра TCCR2 – 001, частота *f=fclk/510*. Для настройки таймера 2 в режим модулятора ШИМ также необходимо задать в регистре TCCR2 единичные значения битов управления PWM2, COM21.

Следовательно, запуск таймера 2 в режиме модулятора ШИМ производится записью байта управления 0b01100001 в регистр TCCR2, а управление выходным напряжением – записью кода управления в регистр OCR2. Необходима также инициализация выхода OC2 (PD7): в регистре DDRD бит 7 должен иметь единичное значение, определяя для PD7 режим вывода.

4.5. Порядок выполнения лабораторной работы

В соответствии с вариантом индивидуального задания подготовить программу обработки аналоговых сигналов, включающую прием аналогового сигнала, преобразование в цифровой код, отображение этого кода цифровым индикатором лабораторного стенда, преобразование кода по заданному алго-

ритму и формирование аналогового выходного сигнала через ЦАП и модулятор ШИМ микроконтроллера. При выполнении лабораторной работы использовать программные модули, подготовленные в лабораторных работах 1 и 2. Необходимо учитывать, что для отображения данных на цифровом индикаторе лабораторного стенда необходимо преобразовать данные из шестнадцатеричного формата в двоично-десятичный формат (см. лабораторную работу № 2).

Измерение параметров аналоговых выходных сигналов необходимо производить осциллографом. Одновременно подавая на два входа осциллографа сигналы ЦАП и модулятора ШИМ, в масштабе зарисовать временные диаграммы напряжений и сравнить характеристики выходных сигналов при разных способах их формирования.

5. ЛАБОРАТОРНАЯ РАБОТА № 4 ПРИМЕНЕНИЕ МИКРОКОНТРОЛЛЕРОВ В РЕАЛИЗАЦИИ АЛГОРИТМОВ УПРАВЛЕНИЯ

5.1. ЦЕЛЬ РАБОТЫ

В работе изучаются вопросы построения комплекса интерфейсов для реализации алгоритмов управления техническими объектами с помощью микроконтроллеров.

5.2. Основные функции системы автоматического управления

Независимо от характера решаемых задач практически любую систему автоматического управления можно представить стандартной структурой, приведенной на рис. 5.1.

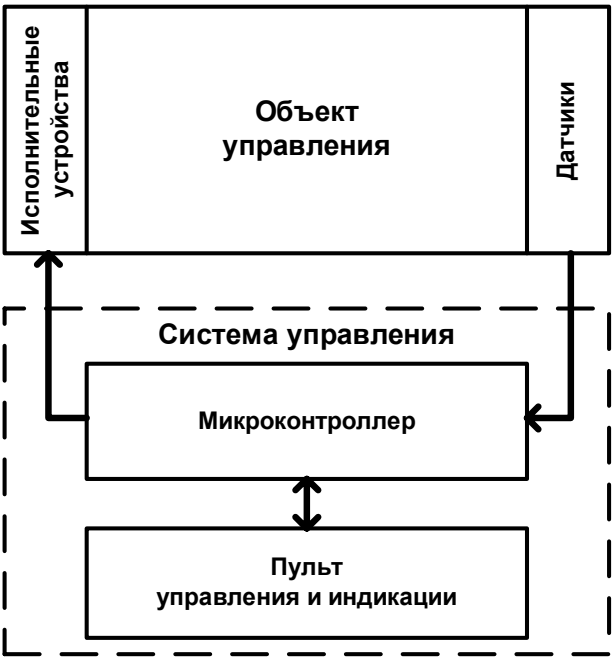

Рис. 5.1. Структурная схема системы управления

Для управления любым объектом необходима информация о его состоянии, которая поступает от датчиков. Микроконтроллер, реализуя основные функции управления, принимает сигналы датчиков, на основе этих данных в соответствии с заданными алгоритмами формирует управляющие воздействия на исполнительные устройства для управления состоянием объекта. Микроконтроллерная система содержит пульт управления и индикации, который позволяет отображать данные о текущих режимах и параметрах системы и вводить управляющие данные для изменения режимов и параметров работы.

Типовая задача управления заключается в стабилизации одного из параметров объекта, например: стабилизация температуры, скорости вращения электропривода и т.п. При решении таких задач необходимо с пульта управления задавать требуемое значение параметра объекта, от датчиков получать текущее значение параметра, определять отклонение параметра (ошибку), в соответствии с ошибкой формировать управляющее воздействие на исполнительные устройства.

В лабораторном стенде ввод требуемого значения параметра можно производить аналоговым сигналом, подавая его на вход АЦП, либо цифровым кодом в параллельном формате через порты микроконтроллера, в последовательном формате через интерфейс UART. Сигналы с датчиков также можно принимать как аналоговые, так и цифровые, используя те же интерфейсы лабораторного стенда.

Цифровые сигналы управления для исполнительных устройств целесообразно передавать в параллельном формате. Аналоговые сигналы управления формируются либо с использованием ЦАП лабораторного стенда, либо таймерами микроконтроллера в режиме модуляторов ШИМ. Работа системы управления, как правило, требует отображения режимов работы и их параметров, необходимые средства отображения входят в состав лабораторного стенда. Вопросы построения интерфейсов для реализации всех этих функций изучались в предыдущих лабораторных работах.

Используя в качестве объекта управления двигатель постоянного тока лабораторного стенда по курсу "Электронные устройства автоматики", можно построить систему стабилизации скорости. Управление скоростью вращения двигателя необходимо производить изменением напряжения якоря. Скорость измеряется таходатчиком, частота выходного сигнала которого пропорциональна скорости. При построении системы стабилизации задачу целесообразно решать в следующей последовательности:

1. Выбрать способ формирования напряжения управления для двигателя. Например, используем в таймере 1 канал А в режиме модулятора ШИМ с 8-битовым кодом управления. Это позволяет обеспечить формирование сигнала управления с дискретностью $1/2^8$=0,4%. Так как мощность вы-

ходного сигнала микроконтроллера недостаточна, необходимо выходной сигнал ШИМ подавать на двигатель постоянного тока через дополнительный, более мощный ключевой каскад. Управляющее напряжение для двигателя также может быть получено с помощью ЦАП лабораторного стенда.

2. Частотный сигнал таходатчика необходимо преобразовать в цифровой код для реализации алгоритма управления. Это преобразование можно выполнить подсчетом числа импульсов таходатчика в течение заданного интервала времени. Интервал времени может задаваться таймером 0, а импульсы таходатчика в течение заданного интервала времени можно подсчитывать таймером 2. Дискретность кода скорости, определяемая разрядностью таймера 2, также равна $1/2^8$=0,4%. Изменяя длительность интервала времени для преобразования, можно производить масштабирование формируемого кода скорости. Код скорости также можно получить с помощью таймера 2 в режиме "захват" (capture).

3. Выбрать способ задания стабилизируемой скорости вращения двигателя. Например, потенциометром лабораторного стенда можно регулировать напряжение на входе АЦП микроконтроллера, а его выходной код после преобразования в 8-битовый формат и масштабирования использовать в качестве кода управления скоростью.

4. Кнопками управления стенда реализовать функции включения/выключения двигателя. На цифровом индикаторе стенда отображать заданное значение скорости вращения и его текущее значение при включенном двигателе.

После определения функций для всех используемых элементов лабораторного стенда необходимо выбрать параметры настройки и инициализации аппаратных средств микроконтроллера, подготовить алгоритм рабочей программы микроконтроллера для решения задачи стабилизации скорости вращения двигателя.

Рабочая программа должна обеспечивать в каждом цикле выполнение следующих функций:

- Прием сигнала таходатчика, формирование и отображение на индикаторе кода текущего значения скорости.
- Формирование и отображение кода заданного значения скорости.
- Вычисление ошибки (отклонения скорости от заданного значения), формирование нового значения выходного кода управления для двигателя с учетом знака и величины ошибки.
- Прием и обработку сигналов кнопок управления для включения/выключения двигателя.

5.3. Порядок выполнения лабораторной работы

В соответствии с вариантом индивидуального задания разработать алгоритм и программу для реализации системы стабилизации. При разработке алгоритма работы необходимо выбрать средства лабораторного стенда и микроконтроллера, которые будут использоваться для решения всех необходимых задач; определить их параметры и режимы, обеспечивая корректное взаимодействие.

Алгоритм должен предусматривать ввод данных для задания требуемых параметров и режимов работы с их отображением элементами индикации стенда, реализацию заданного закона управления и формирование необходимых сигналов для управления исполнительными устройствами. В лабораторной работе рекомендуется использовать программные модули, подготовленные при выполнении лабораторных работах 1 – 3.

Библиографический список

1. Иванов Ю.И., Югай В.Я. Применение микроконтроллеров AVR: Учебное пособие.–Таганрог: Изд-во ТРТУ, 2003.–72 с.

ПРИЛОЖЕНИЕ

Арифметические и логические команды

Команда	Описание	Флаги	Примечание
add Rd, Rr	Сложение, Rd ← Rd+Rr	HVNCZ	
adc Rd, Rr	Сложение с переносом, Rd ← Rd+Rr+C	HVNCZ	
adiw Rd, K	Сложение слова (2 байта, R(d+1)Rd) с константой K, R(d+1)Rd ← R(d+1)Rd+K	SVNCZ	d=(24,26,28,30), K=(0 – 63)
sub Rd, Rr	Вычитание, Rd ← Rd–Rr	HVNCZ	
subi Rd, K	Вычитание константы, Rd ← Rd–K	HVNCZ	d=(16 – 31)
sbc Rd, Rr	Вычитание с переносом, Rd ← Rd–Rr–C	HVNCZ	
sbci Rd, K	Вычитание константы с переносом, Rd ← Rd–K–C	HVNCZ	d=(16 – 31)
sbiw Rd, K	Вычитание из слова (2 байта, R(d+1)Rd) констан-ты K, R(d+1)Rd ← R(d+1)Rd–K	SVNCZ	d=(24,26,28,30), K=(0 – 63)
inc Rd	Инкремент, Rd ← Rd+1	VNZ	
dec Rd	Декремент, Rd ← Rd–1	VNZ	
and Rd, Rr	Логическое И, Rd ← Rd•Rr	VNZ	
andi Rd, K	Логическое И с константой, Rd ← Rd•K	VNZ	d=(16 – 31)
or Rd, Rr	Логич. ИЛИ, Rd ← Rd ∨ Rr	VNZ	
ori Rd, K	Логич. ИЛИ с константой, Rd ← Rd ∨ K	VNZ	d=(16 – 31)
eor Rd, Rr	Исключающее ИЛИ, Rd ← Rd ⊕ Rr	VNZ	

Команда	Описание	Флаги	Примечание
com Rd	Инверсия, Rd ← $ff–Rd	VNZ	
neg Rd	Дополнение, Rd ← $00–Rd	HVNCZ	
asr Rd	Арифметический сдвиг вправо, старший бит Rd(7) не изменяется, остальные биты Rd сдвигаются вправо, флаг C ← Rd(0)	VNCZ	
lsl Rd	Логический сдвиг влево, все биты Rd смещаются влево, младший бит Rd(0) ← 0, флаг C ← Rd(7)	VNCZ	
lsr Rd	Логический сдвиг вправо, все биты Rd смещаются вправо, старший бит Rd(7) ← 0, флаг C← Rd(0)	VNCZ	
rol Rd	Циклический сдвиг влево с переносом, все биты Rd смещаются влево, младший бит Rd(0) ← C, флаг C ← Rd(7)	VNCZ	
ror Rd	Циклический сдвиг вправо с переносом, все биты Rd смещаются вправо, старший бит Rd(7) ← C, флаг C ← Rd(0)	VNCZ	
tst Rd	Тест на ноль или минус Rd не изменяется, флаги Z и N определяются содержимым Rd, V ← 0	SVNZ	

Команда	Описание	Флаги	Примечание
cp Rd, Rr	Сравнение, содержимое регистров не изменяется, флаги формируются в соответствии с результатом вычитания Rd – Rr	HVNCZ	
cpc Rd, Rr	Сравнение с учетом переноса, содержимое регистров не изменяется, флаги формируются в соответствии с результатом вычитания Rd – Rr – C	HVNCZ	
cpi Rd, K	Сравнение с константой, содержимое регистров не изменяет, флаги формируются в соответствии с результатом вычитания Rd – K	HVNCZ	d=(16 – 31)

Команды пересылки данных

Команда	Описание	Флаги	Примечание
mov Rd, Rr	Пересылка данных между регистрами, Rd ← Rr	Нет	
ldi Rd, K	Загрузка константы в регистр, Rd ← K	Нет	d=(16 – 31)
lds Rd, k	Пересылка данных в Rd из ОЗУ (адрес k), Rd ← ОЗУ(k)	Нет	
sts k, Rr	Пересылка данных из Rr в ОЗУ (адрес k), ОЗУ(k) ← Rr	Нет	

Команда	Описание	Флаги	Примечание
in Rd, P	Пересылка данных из регистра ввода-вывода P в регистр Rd, Rd ← P	Нет	
out P, Rr	Пересылка данных из регистра Rr в регистр ввода-вывода P, P ← Rr	Нет	
push Rr	Пересылка данных из регистра Rr в стэк, STACK ← Rr	Нет	
pop Rd	Пересылка данных в регистр Rd из стэка, Rd ← STACK	Нет	
Команды пересылки косвенной адресации с использованием регистров X, Y, Z			
ld Rd, X	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром X, Rd ← ОЗУ(X)	Нет	
ld Rd, –X	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром X, и предшествующим декрементом X, Rd ← ОЗУ(X–1), X ← X–1	Нет	
ld Rd, X+	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром X, и последующим инкрементом X, Rd ← ОЗУ(X), X ← X+1	Нет	
st X, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным регистром X, ОЗУ(X) ← Rr	Нет	

Команда	Описание	Флаги	Примечание
st –X, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным регистром X, и предшествующим декрементом X, ОЗУ(X–1) ← Rr, X ← X–1	Нет	
st X+, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным регистром X, и последующим инкрементом X, ОЗУ(X) ← Rr, X ← X+1	Нет	
ld Rd, Y	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром Y, Rd ← ОЗУ(Y)	Нет	
ld Rd, –Y	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром Y, и предшествующим декрементом Y, Rd ← ОЗУ(Y–1), Y ← Y–1	Нет	
ld Rd, Y+	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром Y, и последующим инкрементом Y, Rd ← ОЗУ(Y), Y ← Y+1	Нет	
ldd Rd, Y+k	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным регистром Y со смещением k, Rd ← ОЗУ(Y+k)	Нет	k=(0 – 63)
st Y, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным регистром Y, ОЗУ(Y) ← Rr	Нет	
st –Y, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Y, и предшествующим декрементом Y, ОЗУ(Y–1) ← Rr, Y ← Y–1	Нет	

Команда	Описание	Флаги	Примечание
st Y+, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Y, и последующим инкрементом Y, ОЗУ(Y) ← Rr, Y ← Y+1	Нет	
std Y+k, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Y со смещением k, ОЗУ(Y+k) ← Rr	Нет	k=(0 – 63)
ld Rd, Z	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным ре-гистром Z. Rd ← ОЗУ(Z)	Нет	
ld Rd, –Z	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным ре-гистром Z, и предшествую-щим декрементом Z Rd ← ОЗУ(Z–1), Z ← Z–1	Нет	
ld Rd, Z+	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным ре-гистром Z, и последующим ин-крементом Z, Rd ← ОЗУ(Z), Z ← Z+1	Нет	
ldd Rd, Z+k	Пересылка данных в регистр Rd из ОЗУ с адресом, указанным ре-гистром Z со смещением k, Rd ← ОЗУ(Z+k)	Нет	k=(0 – 63)
st Z, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Z, ОЗУ(Z) ← Rr	Нет	
st –Z, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Z, и предшествую-щим декрементом Z, ОЗУ(Z–1) ← Rr, Z ← Z–1	Нет	

Команда	Описание	Флаги	Примечание
st Z+, Rr	Пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Z, и последующим инк-рементом Z, ОЗУ(Z) ← Rr, Z ← Z+1	Нет	
std Z+k, Rr	пересылка данных из регистра Rr в ОЗУ с адресом, указанным ре-гистром Z со смещением k, ОЗУ(Z+k) ← Rr	Нет	k=(0 – 63)
lpm	Пересылка данных в регистр R0 из памяти программ (РМ) по адресу, указанному регистром Z, R0 ← PM(Z)	Нет	

Команды управления

Команда	Описание	Флаги	Примечание
rjmp k	Безусловный переход (изме-нение PC на k), PC ← PC+1+k	Нет	–2048 < k < 2048
ijmp	Безусловный переход с ад-ресацией по Z, PC ← Z	Нет	
rcall k	Вызов подпрограммы (изме-нение PC на k), STACK← PC, PC← PC+1+k	Нет	–2048 < k < 2048
icall	Вызов подпрограммы с ад-ресацией по Z, STACK← PC, PC← Z	Нет	
ret	Возврат из подпрограммы, PC ← STACK	Нет	

Команда	Описание		Флаги	Примечание
reti	Возврат из прерывания, PC ← STACK, I = 1		I	
brbs b, k	Условный переход (изменение PC на k), если бит b в SREG установлен, PC ← PC+1+k, если SREG(b)=1		Нет	– 64 ≤ k ≤ 63
brbc b, k	Условный переход (изменение PC на k), если бит b SREG очищен, PC ← PC+1+k, если SREG(b)=0		Нет	– 64 ≤ k ≤ 63
breq k	Условный переход (изменение PC на k), если равно, PC ← PC+1+k, если флаг Z=1		Нет	– 64 ≤ k ≤ 63
brne k	Переход (изменение PC на k), если не равно, PC ← PC+1+k, если флаг Z=0		Нет	– 64 ≤ k ≤ 63
brcs k (brsh k)	Условный переход на k по флагу C=1, PC ← PC+1+k, если флаг C=1		Нет	– 64 ≤ k ≤ 63
brcc k (brlo k)	Условный переход на k по флагу C=0, PC ← PC+1+k, если флаг C=0		Нет	– 64 ≤ k ≤ 63
brmi k	N=1	Условный переход на k по флагу N	Нет	– 64 ≤ k ≤ 63
brpl k	N=0			
brvs k	V=1	Условный переход на k по флагу V	Нет	– 64 ≤ k ≤ 63
brvc k	V=0			
brge k	S=1	Условный переход на	Нет	– 64 ≤ k ≤ 63

brlt k	S=0	k по флагу S		
Команда	Описание		Флаги	Примечание
brhs k	H=1	Условный переход на k по флагу H	Нет	$-64 \le k \le 63$
brhc k	H=0			
brts k	T=1	Условный переход на k по флагу T	Нет	$-64 \le k \le 63$
brtc k	T=0			
brie k	I=1	Условный переход на k по флагу I	Нет	$-64 \le k \le 63$
brid k	I=0			
Следующие команды пропускают одну операцию в программе, если условие выполняется				
sbrs Rr, b	Пропустить, если бит b в регистре Rr установлен, PC ← PC+2(3), если Rr(b) = 1		Нет	
sbrc Rr, b	Пропустить, если бит b в регистре Rr очищен, PC ← PC+2(3), если Rr(b) = 0		Нет	
sbis P, b	Пропустить, если бит b в регистре ввода-вывода P установлен, PC ← PC+2(3), если P(b) = 1		Нет	P = (0 – 31)
sbic P, b	Пропустить, если бит b в регистре ввода-вывода P очищен , PC ← PC+2(3), если P(b) = 0		Нет	P = (0 – 31)
cpse Rd, Rr	Пропустить, если Rd = Rr, PC ← PC+2(3), если Rd = Rr		Нет	

Команды преобразования битов в регистрах

Команда	Описание	Флаги	Примечание
sbr Rd, K	Установить в регистре Rd биты по маске K	Нет	d = (16 – 31)
cbr Rd, K	Очистить в регистре Rd биты по маске K	Нет	d = (16 – 31)
sbi P, b	Установить в регистре ввода-вывода P бит b, P(b)=1	Нет	P = (0 – 31)
cbi P, b	Очистить в регистре ввода-вывода P бит b, P(b)=0	Нет	P = (0 – 31)
bst Rr, b	Пересылка бита b из регистра Rr во флаг T, T ← Rr(b)	T	
bld Rd, b	Пересылка бита b в регистр Rd из флага T, Rr(b) ← T	Нет	
swap Rd	Поменять местами старшую и младшую тетрады в регистре Rd	Нет	
ser Rd	Установить все биты регистра Rd, Rd ← $ff	Нет	d = (16 – 31)
clr Rd	Очистить все биты регистра Rd, Rd ← $00	VNZ	
bset b	Установить бит b в регистре SREG, SREG(b) ← 1	SREG(b)	
bclr b	Очистить бит b в регистре SREG, SREG(b) ← 0	SREG(b)	
sec	Установить флаг C, C ← 1	C	
clc	Очистить флаг C, C ← 0	C	
sez	Установить флаг Z, Z ← 1	Z	
clz	Очистить флаг Z, Z ← 0	Z	
sen	Установить флаг N, N ← 1	N	

cln	Очистить флаг N, N ← 0	N	
sev	Установить флаг V, V ← 1	V	
Команда	Описание	Флаги	Примеч.
clv	Очистить флаг V, V ← 0	V	
ses	Установить флаг S, S ← 1	S	
cls	Очистить флаг S, S ← 0	S	
seh	Установить флаг H, H ← 1	H	
clh	Очистить флаг H, H ← 0	H	
set	Установить флаг T, T ← 1	T	
clt	Очистить флаг T, T ← 0	T	
sei	Установить флаг I, I ← 1	I	
cli	Очистить флаг I, I ← 0	I	

Прочие команды

Команда	Описание	Флаги	Примечание
nop	Нет операции	Нет	
sleep	Переход в режим sleep	Нет	
wdr	Сброс сторожевого таймера	Нет	

В мнемонических обозначениях команд всегда первым указывается регистр, в который помещается результат операции: любой регистр файла регистров общего назначения, файла регистров ввода-вывода, ячейка ОЗУ. Ог-

раничения на параметры и операнды команд приведены в примечании. В описании команд приняты следующие обозначения:

Rd – регистр, в который помещается результат операции (любой регистр общего назначения).

Rr – регистр, из которого поступает байт данных для операции (любой регистр общего назначения).

K – константа (байт данных), число в десятичном формате от 0 до 255, в шестнадцатеричном формате могут использоваться два варианта обозначения: 0x00 – 0xff или $00 – $ff, в двоичном формате –следующее обозначение: 0b00000000 – 0b11111111.

k – константа–адрес (два байта в пределах каждого адресного пространст-ва), также может указываться в десятичном, шестнадцатеричном или двоичном формате.

X, Y, Z – регистры косвенной адресации (в файле регистров общего назначения регистр X – R27,R26; регистр Y – R29,R28; регистр Z – R30,R31).

P – регистр файла регистров ввода-вывода.

b – бит от 0 до 7 в любом регистре (старший бит – 7, младший бит – 0).

PC – программный счетчик.

STACK – стэк.

PM – память программ.

Регистр состояния (флагов) микроконтроллера SREG играет важную роль в рабочих программах, биты этого регистра определяют условия для выполнения команд управления.

Регистр состояния микроконтроллера SREG

I	T	H	S	V	N	Z	C

Флаги регистра SREG (начиная со старшего бита):

I – флаг глобального разрешения прерывания, разрешает (1) или запрещает (0) все аппаратные прерывания.

T – флаг копирования бита, может быть скопирован из любого бита (или в любой бит) любого регистра общего назначения.

H – флаг переноса между младшей и старшей тетрадой байта данных.

S – флаг знака, определяется суммой по mod2 флагов $N \oplus V$.

V – флаг переполнения (дополнения до двух).

N – флаг отрицательного результата (соответствует значению бита 7 результата операции).

Z – флаг нулевого результата операции.

C – флаг переноса.

При использовании команд с анализом флагов регистра SREG необходимо учитывать, что эти флаги формируются не всеми операциями, например, операции пересылки данных флаги не изменяют.

Иванов Юрий Иванович
Югай Владислав Яковлевич

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ СИСТЕМ УПРАВЛЕНИЯ

Ответственный за выпуск Югай В.Я.
Редактор Белова Л.Ф.
Корректор

ЛП №
Подписано к печати
Офсетная печать
Усл. п.л. –3,2 Уч.-изд.л. – 3,0
Заказ №
Тираж экз.
"С"

Издательство Таганрогского государственного радиотехнического университета
ГСП 17А, Таганрог, 28, Некрасовский, 44
Типография Таганрогского государственного радиотехнического университета
ГСП 17А, Таганрог, 28, Энгельса, 4