Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ульяновский государственный технический университет

СПРАВОЧНЫЕ МАТЕРИАЛЫ ПО ПРОГРАММИРОВАНИЮ МИКРОКОНТРОЛЛЕРОВ

Учебно-методические указания к лабораторным работам
для студентов специальности
«Вычислительные машины, комплексы, системы и сети»

Составители: В. Н. Негода
А. В. Лылова
О. В. Ратанова

Ульяновск 2006

УДК 681.3(076)
ББК 32.973-01я7
С74

Рецензент
кандидат технических наук, доцент, декан ФИСТ В. В. Шишкин

Одобрено секцией методических пособий научно-методического совета университета

С74 **Справочные** материалы по программированию микроконтроллеров: учебно-методические указания для студентов специальности «Вычислительные машины, комплексы, системы и сети»/ сост. В. Н. Негода, А. В. Лылова, О. В. Ратанова. – Ульяновск: УлГТУ, 2006. – 46 с.

Указания написаны в соответствии с рабочей программой дисциплины «Микропроцессорные системы» для студентов пятого курса специальности «Вычислительные машины, комплексы, системы и сети». Представлены справочные сведения по архитектуре микроконтроллеров семейств MicroChip PIC17C75x и AVR ATMega для поддержки программирования задач по лабораторным работам и курсовому проектированию.

Подготовлены на кафедре «Вычислительная техника».

УДК 681.3(076)
ББК 32.973-01я7

Содержание

Введение 4
1. Справочные данные по семейству микроконтроллеров PIC17C75x 5
1.1. Общая характеристика семейства 5
1.2. Программно-доступные компоненты и способы адресации 6
1.3. Форматы команд и способы адресации 18
1.4. Система команд 20
2. Справочные данные по архитектуре микроконтроллеров AVR ATMega 24
2.1. Общая характеристика семейства ATMega 24
2.2. Программно-доступные компоненты и способы адресации 24
2.3. Форматы команд и способы адресации 35
2.4. Система команд 37
Библиографический список 46

Введение

Доминирующим видом проектных работ, выполняемых при создании микропроцессорных систем, является программирование микроконтроллеров (МК). Разработка аппаратной части зачастую сводится к выбору готовых плат контроллеров, имеющих в своем составе нужную номенклатуру каналов сопряжения с объектом контроля и управления. Однако при поиске в Интернет справочных материалов, связанных с микроконтроллерами, обнаруживается, что подавляющее число доступных источников представляют собой англоязычные технические описания, предназначенные, прежде всего, для проектирования аппаратных средств. Много реже встречаются руководства для программистов, причем объем таких руководств обычно составляет несколько сотен страниц.

Учебно-инженерные задачи для лабораторных работ и курсового проектирования по учебной дисциплине «Микропроцессорные системы» целесообразно строить таким образом, чтобы студенты могли познакомиться с архитектурой различных микроконтроллеров. Поскольку учебное время очень ограничено, делать ставку на англоязычные описания нерационально. Это могут себе позволить только те студенты, которые решили специализироваться именно в проектировании микропроцессоных систем. Русскоязычные технические документы и руководства являются библиографической редкостью и имеют довольно большой объем, причем зачастую содержат слишком много информации, не связанной с основным видом работы студентов – программированием.

Все вышесказанное обусловливает необходимость составлять достаточно компактные справочники по архитектуре различных микроконтроллеров, ориентированные на использование в учебном процессе. В данных учебно-методических материалах приводятся основные справочные сведения по двум микроконтроллерным архитектурам – семейству микроконтроллеров фирмы MicroChip PIC17C75x и семейству микроконтроллеров фирмы Atmel AVR ATMega. Микроконтроллеры этих семейств в последние годы наиболее динамично расширяли долю своего присутствия на рынке электронных компонентов для микропроцессорных систем. Это дает основание надеяться, что конкретные знания этих микроконтроллерных архитектур позволят выпускникам легче адаптироваться к требованиям рынка труда.

1. Справочные данные по семейству микроконтроллеров PIC17C75x

1.1. Общая характеристика семейства

Существует несколько семейств МК PICMicro. Справочные данные этого раздела методических указаний сформированы на основе описаний, приведенных в книге [1]. Это старшие модели архитектуры МК, процессорное ядро которых базируется на следующих основных архитектурных решениях:

- сочетание RISC архитектуры и традиционных решений одноаккумуляторной архитектуры с 8-разрядным арифметико-логическим устройством (АЛУ);
- раздельное адресное пространство программ и данных с возможностью переносить данные из одного адресного пространства в другое;
- одноуровневая регистровая память данных, включающая в себя регистры общего и специального назначения;
- сочетание 8-разрядных операндов и 16-разрядных команд при доминировании 8-разрядных;
- двухуровневая конвейерная обработка, совмещающая выполнение текущей команды с выборкой следующей;
- компактный набор из 58 команд, включающей в себя аппаратное умножение, занимающее один машинный такт;
- возможность программировать программное ППЗУ как вне микропроцессорного устройства, так и без демонтажа микроконтроллера из устройства;
- широкий спектр встроенных периферийных устройств и средств сопряжения с внешними устройствами, включающий в себя: 50 линий ввода-вывода, 18 источников прерываний, 4 таймера-счетчика, 4 входа захвата данных, 3 выходных широтно-импульсных модулятора (ШИМ), 2 универсальных синхронно-асинхронных приемопередатчика USART, 10-битный 12-канальный АЦП, синхронный последовательный порт SSP с SPI и I2C;
- наличие режимов энергосбережения, сторожевого таймера и защиты доступа к памяти программ.

Первые пять из перечисленных архитектурных решений дают возможность иметь одноадресные двухоперандные команды с достаточно длинным полем

адреса, выполняемые за один машинный цикл (4 такта задающего генератора). Поскольку выборка следующей команды в большинстве случаев выполняется на фоне выполнения текущей, то производительность МК близка к $F/4$ команд в секунду, где F – частота задающего генератора, предельное значение которой для PIC17C75x равно 33 МГц.

1.2. Программно-доступные компоненты и способы адресации

Основными программно-доступными компонентами PIC17C75x, задаваемыми в аргументах командных строк ассемблер-программы, являются:

- внутренняя память программ (8/16 Кбайт для PIC17C752/PIC17C756);
- внешняя память программ до 64K 16-разрядных слов;
- внутренняя память данных (454/902 байт для PIC17C752/PIC17C756), включающая в себя регистры общего назначения (РгОН) и регистры специального назначения (РгСН);
- рабочий регистр WREG (регистр-аккумулятор), входящий в состав РгСН и неявно адресуемый в большинстве команд обработки данных;
- два адресных регистра FSR0 и FSR1, входящие в состав РгСН и поддерживающие косвенно-регистровую, автоинкрементную и автодекрементную адресации;
- регистр состояния АЛУ ALUSTA, в котором фиксируются признаки результата и задаются режимы косвенно-регистровой адресации через FSR0 и FSR1;
- аппаратный стек 16 двухбайтных слов для хранения адресов возврата из подпрограмм и прерываний;
- двухбайтный счетчик команд PC, младший байт которого PCL входит в адресное пространство РгСН, а старший адресуется неявно;
- регистры конфигурации, входящие в адресное пространство памяти команд;
- регистры управления микроконтроллером;
- регистры встроенных периферийных устройств и портов ввода-вывода.

В адресном пространстве команд первые восемь байтов зарезервированы для стартового фрагмента программы, выполняемой при сбросе МК, а следующие четыре восьмибайтовых сегмента зарезервированы для стартовых фрагментов четырех обработчиков прерываний: от входа RA0/INT – адрес 0008h, приоритет 1 (высший); от таймера TMR0 – адрес 0010h, приоритет 2; от входа –

адрес 0018h, приоритет 3; от периферийных устройств – адрес 0020h, приоритет 4 (низший).

В программной памяти, начинающейся с адреса FE00h, находятся еще три специальные области: FE00h..FE0Fh – слова параметров конфигурации, FE10h..FF5Fh – тестовое ППЗУ, FF60h..FFFFh – загрузочное ПЗУ.

Адрес регистра в машинных командах занимает один байт. Адресное пространство регистров объемом более 256 байт образуется за счет размещения части РгОН и РгСН в различных банках памяти. Номер банка представляется в специальном регистре BSR. Младший полубайт BSR адресует банки РгСН, а старший – банки РгОН. Адресное пространство данных делится на 4 части: 00h..19h – РгСН, видимые из любого банка; 1Ah..1Fh – РгОН, видимые из любого банка; 10h..17h – 8 банков РгСН, выбор которых определяется комбинацией 8-битного исполнительного адреса и BSR[3..0]; 20h..FFh – 2 банка РгОН для PIC17C752 или 4 банка для PIC17C756.

Ниже приводится список РгСН с указанием адресов, идентификаторов, используемых в ассемблер-программах, описания назначения этих регистров и их отдельных разрядов, если это необходимо.

00h: INDF0 Использует значение FSR0 для косвенной адресации регистра памяти данных (физически отсутствует). Фигурирование этого регистра в команде задает использование FSR0 в качестве номера регистра, где находится адрес операнда. Битами ALUSTA[FS1..FS0] может быть задан автоинкремент или автодекремент регистра FSR0 после его использования.

01h: FSR0 Регистр 0 косвенной адресации.

02h: PCL Младший байт PC.

03h: PCLATH Регистр хранения старшего байта PC.

04h: ALUSTA Регистр состояния АЛУ, имеющий внутреннюю структуру <FS3,FS2,FS1,FS0,OV,Z,DC,C>. Разряды: C – признак переноса; DC – признак десятичного переноса; Z – признак нуля; OV – признак переполнения OV; FS1..FS0 – режим косвенной адресации через FSR0 (00 – автоинкремент после обращения, 01 – автодекремент после обращения, 1x – FSR0 не изменяется; FS3..FS2 – режим косвенной адресации через FSR1 (кодировка аналогична FS1,FS0).

05h: TOSTA Регистр управления таймером/счетчиком TMR0. Разряды: 4..1[PS3..PS0] – коэффициент деления предделителя из диапа-

зона {1..min(256,2[PS3..PS0])}; 6..5[TOSE..TOCS] – выбор условия приращения TMR0 (00 – по заднему фронту сигнала на входе RA1/TOCKI, 10 – по переднему фронту, *1 – по импульсу внутреннего генератора); 7[INTEDG] – выбор условия прерывания на входе RA0/INT (1 – по переднему фронту, 0 – по заднему фронту).

06h: CPUSTA Регистр состояния процессора. Разряды: 0 [-BOR] – информирование о сбросе по снижению напряжения питания (1 – сброс не происходил, 0 – сброс прозошел); 1[-POR] – информирование о сбросе после включения питания (1 – сброс не происходил, 0 – сброс произошел); 2 [-PD] – информирование о причине включения питания (1 – после POR или команды CLRWDT, 0 – после выхода из режима SLEEP); 3 [-TO] – информирование о причине тайм-аута сторожевого таймера WDT (1 – после POR или команды CLRWDT, 0 – после тайм-аута WDT); 4 [GLINTD] – бит общего запрещения прерывания (1 – запрещены, 0 – разрешены); 5 [STKAV] – признак доступности стека (1 – стек доступен, 0 – стек переполнен).

07h: INTSTA Регистр статуса прерываний. Разряды: 0 [INTE] – разрешение прерывания по входу INT; 1 [TOIE] – разрешение прерывания по переполнению TMR0; 2 [TOCKIE] – разрешение прерывания по входу TOCKI; 3 [PEIE] – разрешение прерываний от периферийных устройств, биты индивидуального разрешения которых установлены в 1; 4 [INTF] – флаг прерывания по входу INT, очищается аппаратно при прерывании по вектору 08h; 5 [TOIF] – флаг прерывания по переполнению TMR0, очищается аппаратно при прерывании по вектору 10h; 6 [TOCKIF] – флаг прерывания по входу TOCKI, очищается аппаратно при прерывании по вектору 18h; 7 [PEIF] – флаг периферийного прерывания.

08h: INDF1 Использует значение FSR1 для косвенной адресации регистра памяти данных (физически отсутствует). Фигурирование этого регистра в команде задает использование FSR1 в качестве номера регистра, где находится адрес операнда. Битами ALUSTA[FS3..FS2] может быть задан автоинкремент или автодекремент регистра FSR1 после его использования.

09h: FSR1 Регистр 1 косвенной адресации.

0Ah: WREG Рабочий регистр (регистр-аккумулятор).

0Bh: TMR0L Младший байт счетчика-таймера TMR0. Программно-недоступный предделитель таймера TMR0 образует истинный нечитаемый младший байт 24-разрядного счетчика. В этой связи TMR0L можно считать средним байтом. Предделитель сбрасывается в 0 при записи в TMR0.

0Ch: TMR0H] Старший байт счетчика-таймера TMR0.

0Dh: TBLPTRL Младший байт указателя памяти программ.

0Eh: TBLPTRH Старший байт указателя памяти программ.

0Fh: BSR Регистр выбора банка регистровой памяти. Разряды: 3..0 – номер банка регистров специального назначения для диапазона адресов 10h..17h; 7..4 – номер банка регистров общего назначения для диапазона адресов 20h..FFh.

18h: PRODL Младший байт произведения на выходе умножителя.

19h: PRODH Старший байт произведения на выходе умножителя.

Банк 0

10h: PORTA Порт А и разряды обслуживания другой периферии. Разряды: 0 [RA0/INT] – вход PORTA[0]/вход внешнего прерывания; 1 [RA1/TOCKI] – вход PORTA[1]/вход внешнего прерывания и/или вход тактового сигнала для TMR0; 2 [RA2/-SS/SSL] – вход или выход с открытым стоком PORTA[2]/вход выбора ведомого SPI или тактовый сигнал I^2C; 3 [RA3/SDI/SDA] – вход или выход с открытым стоком PORTA[3]/вход данных SPI или канал данных I^2C; 4 [RA4/RX/DT] – вход или выход PORTA[4]/вход приемника USART1/данные SCI1; 5 [RA5/TX/CK] – вход или выход PORTA[5]/выход передатчика USART1/тактовый сигнал SCI1; 7 [-RBPU] – бит управления подтягивающими резисторами PORTB.

11h: DDRB Регистр направления данных PORTB. 1 настраивает соответствующий канал на ввод, а 0 – на вывод.

12h: PORTB Двунаправленный 8-разрядный порт данных PORTB. Чтение из PORTB возвращает состояние соответствующих выводов МК, запись в PORTB фиксирует байт в защелке. Для включения подтягивающих резисторов необходимо занести 0 в PORTA[7]. При PIE1[RBIE]=1 несовпадение сигналов на входах порта со значением в защелке PORTB инициирует прерывание и взво-

дит флаг PIR1[RBIF]. Разряды: 0 [RB0/CAP1] – вход-выход/вход регистра захвата CAP1; 1 [RB1/CAP2] – вход-выход/вход регистра захвата CAP2; 2 [RB2/PWM1] – вход-выход/выход ШИМ PWM1; 3 [RB3/PWM2] – вход-выход/выход ШИМ PWM2; 4 [RB4/TCLK12] – вход-выход/вход внешнего тактового сигнала для TMR1 и TMR2; 5 [RB5/TCLK3] – вход-выход/вход внешнего тактового сигнала для TMR3; 6 [RB6/SCK] – вход-выход/линия тактового сигнала SPI; 7 [RB7/SDO] – вход-выход/выход данных SPI.

13h: RCSTA1 Регистр состояния-управления приемником UART1. Разряды: 0 [RX9] – 9-й бит принятых данных; 1 [OERR] – признак ошибки переполнения буфера; 2 [FERR] – признак ошибки кадра; 3 – не реализован; 4 [CREN] – разрешение поточного приема в синхронном режиме; 5 [SREN] – разрешение поточного приема в асинхронном режиме; 6 [RX9] – разрешение 9-битного приема; 7 [SPEN] – разрешение работы последовательного порта.

14h: RCREG1 Регистр приемника последовательного порта 1.

15h: TXSTA1 Регистр состояния-управления передатчиком UART1. Разряды: 0 [RX9] – 9-й бит передаваемых данных; 1 [TRMT] – признак незаполненности сдвигового регистра передатчика TSR; 3..2 – не реализованы; 4 [SYNC] – выбор режиа работы (1 – синхронный, 0 – асинхронный); 5 [TXEN] – разрешение передачи; 6 [TX9] – разрешение 9-битной передачи; 7 [CSRC] –выбор источника тактового сигнала в синхронном режиме (1 – ведущий, внутренний тактовый сигнал от BRG, 0 – ведомый, внешний тактовый сигнал со входа CK).

16h: TXREG1 Регистр передатчика последовательного порта 1.

17h: SPBRG1 Регистр настройки скорости обмена последовательного порта USART1.

Банк 1

10h: DDRC Регистр направления данных порта PORTC. 1 настраивает на ввод, 0 – на вывод.

11h: PORTC Двунаправленный 8-разрядный порт PORTC, входы/выходы RC7..RC0 которого совмещены с линиями AD7..AD0 систем-

ной шины адреса-данных для подключения внешней памяти программ.

12h: DDRD — Регистр направления данных PORTD. 1 настраивает на ввод, 0 – на вывод.

13h: PORTD — Двунаправленный 8-разрядный порт PORTD, входы/выходы RD7..RD0 которого совмещены с линиями AD15..AD8 системной шины адреса-данных для подключения внешней памяти программ.

14h: DDRE — Регистр направления данных PORTE. 1 настраивает на ввод, 0 – на вывод.

15h: PORTE — Четырехразрядный двунаправленный порт PORTE. Разряды: 0 [RE0/ALE] – вход-выход/ сигнал ALE системной шины; 1 [RE1/-OE] –вход-выход/ сигнал -OE системной шины; 2 [RE2/-WR] – вход-выход/ сигнал -WR системной шины; 3 [RE3/CAP4] – вход-выход/ вход регистра захвата 4.

16h: PIR1 — Регистр флагов периферийных прерываний. Разряды: 0 [RC1IF] – флаг прерывания по переполнению буфера приемника USART1; 1 [TX1IF] – флаг прерывания по опустошению буфера приемника USART1; 2 [CA1IF] – флаг прерывания по захвату данных CAP1; 3 [CA2IF] – флаг прерывания по захвату данных CAP2; 4 [TMR1IF] – флаг прерывания от TMR1; 5 [TMR2IF] – флаг прерывания от TMR2; 6 [TMR3IF] флаг прерывания от TMR3; 7 [RBIF] – флаг прерывания по изменению PORTB.

17h: PIE1 — Регистр индивидуальных разрешений прерываний от периферии. Разряды: 0 [RC1IE] – разрешение прерывания по переполнению буфера приемника USART1; 1 [TX1IE] – разрешение прерывания по опустошению буфера приемника USART1; 2 [CA1IE] – разрешение прерывания по захвату данных CAP1; 3 [CA2IE] – разрешение прерывания по захвату данных CAP2; 4 [TMR1IE] – разрешение прерывания от TMR1; 5 [TMR2IE – разрешение прерывания от TMR2; 6 [TMR3IE] – разрешение прерывания от TMR3; 7 [RBIE] – разрешение прерывания по изменению PORTB.

Банк 2

10h: TMR1 — Таймер-счетчик TMR1.

11h: TMR2 Таймер-счетчик TMR2.

12h: TMR3L Младший байт таймера-счетчика TMR3.

13h: TMR3H Старший байт таймера-счетчика TMR3.

14h: PR1 Период TMR1.

15h: PR2 Период TMR2.

16h: PR3L/CA1L Младший байт периода TMR3/регистра захвата 1.

17h: PR3H/CA1H Старший байт периода TMR3/регистра захвата 1.

Банк 3

10h: PW1DCL Младшие разряды регистра скважности ШИМ1. Разряды: 0..5 – не используются; 7..6 [DC1,DC0] – младшие биты 10-битного значения скважности сигнала ШИМ3.

11h: PW2DCL Младшие разряды регистра скважности ШИМ2 и управление источником значения его периода. Разряды: 0..4 – не используются; 5 [TM2PW2] – 1 назначает в качестве периода ШИМ3 TMR2.PR2, 0 назначает TMR1.PR1; 7..6 [DC1,DC0] – младшие биты 10-битного значения скважности сигнала ШИМ2..

12h: PW1DCH Старшие разряды регистра скважности ШИМ1.

13h: PW2DCH Старшие разряды регистра скважности ШИМ2.

14h: CA2L Младший байт регистра захвата 2.

15h: CA2H Старший байт регистра захвата 2.16h: TCON1 Регистр управления таймерами и регистрами захвата. Разряды: 0 [TMR1CS] – выбор источника тактового сигнала для TMR1 (1 – приращение по заднему фронту сигнала со входа RB4/TCLK12, 0 – TMR3 внутреннего источника сигнала); 1 [TMR2CS] – выбор источника тактового сигнала для TMR2 (1 – приращение по заднему фронту сигнала со входа RB4/TCLK12, 0 – TMR3 внутреннего источника сигнала); 2 [TMR3CS] – выбор источника тактового сигнала для TMR3 (1 – приращение по заднему фронту сигнала со входа RB5/TCLK3, 0 – TMR3 внутреннего источника сигнала); 3 [T16] – выбор режима работы TMR1 и TMR2 (1 – работают совместно как 16-битный таймер, 0 – как два 8-разрядных таймера); 5..4 [CA1ED1,CA1ED0] – выбор события захвата для регистра захвата 1 (00 – по каждому переднему фронту, 01 – по каждому заднему фронту, 10 – по каждому 4-му переднему фронту, 11 – по каждому 16-му переднему фронту);

7..6 [CA2ED1,CA2ED0] – выбор события захвата для регистра захвата 2 (аналогично 5..4);

17h: TCON2 Второй регистр управления таймерами и регистрами захвата. Разряды: 0 [TMR1ON] – бит включения TMR1 или TMR2.TMR1; 1 [TMR2ON] – бит включения TMR2 (должен быть установлен в режиме 16-разрядного таймера TMR2.TMR1; 2 [TMR3ON] – бит включения TMR3; 3 [CA1/-PR3] – бит выбора регистра захвата 1 или регистра периода для TMR3 (1 – CA1 включен, TMR3 работает без регистра периода, 0 – *регистр периода включен, CA1 выключен*); 4 *[PWM1ON]* – бит включения ШИМ1; 5 [PWM2ON] – бит включения ШИМ2; 6 [CA1OVF] – флаг переполнения регистра захвата 1, показывает, что значение CA1H.CA1L не было считано до того, как возникло очередное событие, последующие события захвата не будут обновлять содержимое регистра до тех пор, пока оба его байта не будут считаны; 7 [CA2OVF] – флаг переполнения регистра захвата 2.

Банк 4

10h: PIR2 Регистр флагов периферийных прерываний. Разряды: 0 [RC2IF] – флаг прерывания по переполнению буфера приемника USART2; 1 [TX2IF] – флаг прерывания по опустошению буфера приемника USART1; 2 [CA3IF] – флаг прерывания по захвату данных CAP3; 3 [CA4IF] – флаг прерывания по захвату данных CAP4; 4 – не реализован; 5 [ADIF] – флаг прерывания по завершению преобразования в АЦП; 6 [BCLIF] – флаг прерывания от SSP по конфликту шины I^2C; 7 [SSPIF] – флаг прерывани от SSP.

11h: PIE2 Регистр индивидуальных разрешений периферийных прерываний. Разряды: 0 [RC2IE] – разрешение прерывания по переполнению буфера приемника USART2; 1 [TX2IE] – разрешение прерывания по опустошению буфера приемника USART1; 2 [CA3IE] – разрешение прерывания по захвату данных CAP3; 3 [CA4IE] – разрешение прерывания по захвату данных CAP4; 4 – не реализован; 5 [ADIE] – разрешение прерывания по завершению преобразования в АЦП; 6 [BCLIE] – разрешение прерывания от SSP по конфликту шины I2C; 7 [SSPIE] – разрешение прерывани SSP.

12h Не используется.

13h: RCSTA2 Регистр состояния-управления приемника USART2. Аналогичен RCSTA1.

14h: RCREG2 Регистр приемника USART2.

15h: TXSTA2 Регистр статуса передатчика USART2. Аналогичен TXSTA1.

16h: TXREG2 Регистр передатчика USART2.

17h: SPBRG2 Регистр настройки скорости обмена USART2.

Банк 5

10h: DDRF Регистр направления данных PORTF. 1 настраивает на ввод, 0 – на вывод.

11h: PORTF Двунаправленный 8-разрядный порт PORTF. Каналы RF0..RF1 совмещены с аналоговыми входами AN4..AN11.

12h: DDRG Регистр направления данных PORTG. 1 настраивает на ввод, 0 – на вывод.

13h: PORTG Двунаправленный 8-разрядный порт PORTF. Разряды: 0 [RG0/AN3] – вход-выход/ аналогоивый вход 3; 1 [RG1/AN2] – вход-выход/ аналогоивый вход 2; 2 [RG2/AN1] – вход-выход/ аналогоивый вход 1; 3 [RG3/AN0] – вход-выход/ аналогоивый вход 0; 4 [RG4/CAP3] – вход-выход/ вход регистра захвата 3; 5 [RG5/PWM3] – вход-выход/ выход ШИМ 3; 6 [RG6] – вход-выход; 7 [RG7] – вход-выход.

14h: ADCON0 Регистр управления АЦП. Разряды: 0 [ADON] – бит включения модуля АЦП; 1,3 – не реализованы; 2 [GO/-DONE] – бит статуса преобразования (1 – производится преобразование, запущенное установкой этого бита, 0 – преобразование не производится); 7..4 [CHS3..CHS0] – номер выбираемого аналогового канала;

15h: ADCON1 Регистр управления АЦП. Разряды: 0 [PCFGO] – выбор источника опорного напряжения (1 – напряжение берется со входов V_{REF+} и V_{REF-}; 0 – напряжение берется с выводов AV_{DD} и AV_{SS}; 3..1 [PCFG3..PCFG1] – биты управления настройкой аналог/цифра (A/D) 12 линий AN11..AN0 (000 – все A; 001 – AN7:D, остальные A; 010 – AN7,AN6:D, остальные A; 011 – AN7..AN5:D, остальные A; 100 – AN7..AN4:D, остальные A; 101 – AN11,AN7..AN3:D, остальные A; 110 – AN11,AN10,AN7..AN2:D, остальные A; 111 – все A); 5 [ADFM] – выбор формата результата формирования (1 – выравнивание

вправо, 6 старших разрядов ADRESH читаются как 0; 0 – выравнивание влево, 6 младших разрядов ADRESL читаются как 0); 7..6 [ADCS1..ADCS0] – биты выбора источника тактового сигнала для преобразования (00 – $F_{OSC}/8$, 01 – $F_{OSC}/16$, 10 – $F_{OSC}/64$, 11 – тактовый сигнал F_{RC} от встроенного RC-генератора).

16h: ADRESL Младший байт результата АЦП.

17h: ADRESH Старший байт результата АЦП.

Банк 6

10h: SSPADD Регистр адреса ведомого интерфейса I^2C регистр генератора обмена ведущего I^2C.

11h: SSPCON1 Регистр конфигурирования синхронного порта SSP. Разряды: 3..0 [SSPM3..SSPM0] – режим работы SSP (0000 – ведущий SPI, тактовый сигнал = $F_{OSC}/4$; 0001 – ведущий SPI, тактовый сигнал = $F_{OSC}/16$; 0010 – ведущий SPI, тактовый сигнал = $F_{OSC}/64$; 0011 – ведущий SPI, тактовый сигнал = выход TMR2/2; 0100 – ведомый SPI, тактовый сигнал со входа SCK, управление со входа -SS включено; 0101 – ведомый SPI, тактовый сигнал со входа SCK, управление со входа -SS выключено; 0110 – ведомый I^2C с 7-битным адресом; 0111 – ведомый I^2C с 10-битным адресом; 1000 – ведущий I^2C, тактовый сигнал = $F_{OSC}/(4*SSPADD + 4)$); 4 [CKP] – выбор полярности тактового сигнала (в режиме SPI: 1 – пассивный уровень высокий, 0 – пассивный уровень низкий; в режиме ведомого I^2C: 1 – отпустить тактовый сигнал, 0 – удерживать тактовый сигнал в 0, растягивая его на время подготовки данных); 5 [SSPEN] – включение SSP (1 – в режиме SPI настроить выводы SCK, SDO и SDI, а в режиме I^2C настроить выводы SDA и SCL для работы с последовательным портом; 0 – выключить SSP и настроить его выводы для работы в качестве обычных каналов ввода-вывода); 6 [SSPOV] – бит переполнения приемника (1 – был принят новый байт в то время, пока в SSPBUF содержался предыдущий принятый; 0 – переполнения не было); 7 [WCOL] бит конфликта записи в SSPBUF (1 – была попытка записи в SSPBUF в момент передачи предыдущего байта

или в момент выдачи на шину управляющих битов; 0 – конфликта не было).

12h: SSPCON2 Регистр конфигурирования синхронного порта SSP. Разряды: 0 [SEN] – бит включения START в режиме ведущего I2C, автоматически устанавливается в 0; если модуль I2C занят, этот бит не может быть установлен в 1, автоматически устанавливается в 0 (1 – инициировать START на линиях SCK и SDA, 0 – START не выдается); 1 [SEN] – бит включения повторного START в режиме ведущего I2C, автоматически устанавливается в 0; если модуль I2C занят, этот бит не может быть установлен в 1, автоматически устанавливается в 0 (1 – инициировать повторный START на линиях SCK и SDA, 0 – повторный START не выдается); 2 [PEN] – бит включения STOP в режиме ведущего I2C, автоматически устанавливается в 0; если модуль I2C занят, этот бит не может быть установлен в 1 (1 – инициировать STOP на линиях SCK и SDA, 0 – STOP не выдается); 3 [RCEN] – бит включения повторного приема в режиме ведущего I2C; 4 [ACKEN] – бит включения подтверждения в режиме ведущего I2C, автоматически устанавливается в 0; если модуль I2C занят, этот бит не может быть установлен в 1 (1 – инициировать подтверждение на линиях SCK и SDA, выдать на SDA содержимое бита ACKDT, 0 – подтверждение на SDA не выдается); 5 [ACKDT] – данные для подтверждения принятого байта в режиме ведущего I2C (1 – не выдавать подтверждение, 0 – выдать подтверждение); 6 [ACKSTAT] – бит подтверждения последнего переданного байта в режиме ведущего I2C (1 – подтверждение не было получено, 0 – было получено подтверждение);7 [GCEN] – разрешение общего вызова в режиме ведомого I2C (1 – разрешить прерывание при обнаружении адреса общего вызова, 0 – общий вызов запрещен).

13h: SSPSTAT Регистр состояния-управления синхронного последовательного порта SSP. Разряды: 0 [BF] – флаг заполненности буфера (0 – прием не завершен или передача завершена, 1 – прием завершен, передача не завершена); 1 [UA] – требование обновить адрес для ведомого I^2C с 10-битным адресом; 2 [R/-W] – информация о типе операции в режиме I^2C, полученная вслед за последним совпадением адреса (1 – чтение, 0 – запись);

3 [S] – признак обнаружения бита START в режиме I^2C; 4 [P] – признак обнаружения бита STOP в режиме I^2C; 5 [D/-A] – признак того, что последний принятый байт есть данные, а не адрес; 6 [CKE] – выбор фронта SCK в SPI, по которому передаются данные (задний фронт, если CKE=SSPCON1[CKP]); 7 [SMP] – фаза опроса SPI (для ведущего SPI: 1 – опрос входа в конце периода вывода данных, 0 – в середине периода вывода данных; для I^2C: 1 – управление длительностью фронта включено в стандартном режиме 100 кГц и 1 МГц, 0 – в скоростном режиме 400 кГц).

14h: SSPBUF Буфер данных интерфейса SSP.

15h..17h: Не используются.

Банк 7

10h: PW3DCL Младшие разряды регистра скважности ШИМ3 и управление источником значения его периода. Разряды: 0..4 – не используются; 5 [TM2PW3] 1 назначает в качестве периода ШИМ3 TMR2.PR2, 0 назначает TMR1.PR1; 7..6 [DC1.DC0] – младшие биты 10-битного значения скважности сигнала ШИМ3.

11h: PW3DCH Старшие разряды 10-битного значения скважности сигнала ШИМ3.

12h: CA3L Младший байт регистра захвата 3.

13h: CA3H Старший байт регистра захвата 3.

14h: CA4L Младший байт регистра захвата 4.

15h: CA4H Старший байт регистра захвата 4.

16h: TCON3 Третий регистр управления таймерами и регистрами захвата. Разряды: 0 [PWM3ON] – бит включения ШИМ3; 2..1 [CA3ED1..CA3ED0] – выбор события захвата 3 (00 – по каждому переднему фронту, 01 – по каждому заднему фронту, 10 – по каждому 4-му переднему фронту, 11 – по каждому 16-му переднему фронту); 4..3 [CA4ED1..CA4ED0] – выбор события захвата 4 (аналогично 2..1); 5 [CA3OVF] – флаг переполнения регистра захвата 3, показывает, что значение CA3H.CA3L не было считано до того, как возникло очередное событие, последующие события захвата не будут обновлять содержимое регистра до тех пор, пока оба его байта не будут считаны; 6 [CA4OVF] – флаг переполнения регистра захвата 4.

17h: Не используется.

1.3. Форматы команд и способы адресации

Процессорное ядро МК PIC17C75x реализует следующие основные форматы команд:

ccccccccdffffffff	байт- ориентированные операции с регистрами, где *c..c – биты кода операции, f..f – биты адреса регистра*-операнда, d – бит, определяющий приемник результата, либо модификатор операции;
cccccccckkkkkkkk	операции с 8-битными константами, где k..k – биты константы; в некоторых командах используется четырехбитная константа, размещаемая в младшем или старшем полубайте младшего байта команды;
ccccbbbfffffffff	бит-ориентированные операции с регистрами, где bbb – трехбитный номер адресуемого бита;
ccaaaaaaaaaaaaaa	команды CALL и GOTO, где a..a – 13-разрядный адрес перехода;
cccppppffffffff	двухоперандные операции пересылки байта, минуя WREG, где p..p – 5-разрядный адрес периферийного регистра от 00 до 1Fh;
cccccctiffffffff	операции пересылки, поддерживающие обмен данными с памятью программ, где t – бит, нулевое значение которого задает выбор младшего (TBLATL), а единичное – старшего (TBLATH) байта 16-разрядного буфера данных TBLAT, i – бит, единичное значение *которого задает автоувеличение адреса программной* памяти TBLPTR на единицу после его использования в операции пересылки.

Способы адресации, реализованные в ядре МК, описываются ниже.

Прямая регистровая. В команде указан 8-или 5-битный номер регистра. В операторе ассемблера задается чаще всего имя регистра, имя метки в директиве распределения памяти, хотя возможно и использование значения адреса.

Косвенная адресация памяти данных. В команде фигурирует номер *INDF0/INDF1*, который задает режим использования содержимого *FSR0/FSR1* в качестве номера регистра, где находится операнд. При ALUSTA[FS1]/ALUSTA[FS3] == 1 содержимое FSR0/FSR1 не изменяется, ина-

че после использования регистр косвенной адресации инкрементируется (ALUSTA[FS0]/ALUSTA[FS2] == 1) либо декрементируется (ALUSTA[FS0]/ALUSTA[FS2] == 0). Фигурирование в операторе ассемблера имени FSR0 или FSR1 задает прямую адресацию, а имени INDF0 или INDF1 – косвенную.

Битовая адресация. В адресуемом операнде выполняется операция с указанным через поле bbb разрядом. В операторе ассемблера фигурирует пара операндов: имя или адрес регистра, число или имя константы, значения которых должны находится в пределах 0..7.

Непосредственная адресация. В младшем байте команды присутствует константа, используемая в качестве операнда. В операторе ассемблера фигурирует число или имя константы.

Прямая адресация перехода. В командах перехода CALL и GOTO фигурирует 13-разрядный прямой адрес перехода. Три старших разряда 16-битного адреса при этом берутся из PC[15..13]. В команде перехода к подпрограмме LCALL фигурирует 8-разрядный адрес перехода. Старший байт адреса перехода при этом берется из PCLATH. В операторе ассемблера обычно указывается метка перехода.

Стековая адресация. Реализуется через программно-недоступный указатель аппаратного стека, вмещающего 16 двухбайтовых адресов возврата. Размещение адреса возврата в стеке выполняется при входе в подпрограмму или в обработчик прерываний. Извлечение адреса выполняется по соответствующим командам возврата.

Косвенная адресация памяти программ. Содержимое двухбайтного регистра – указателя программной памяти TBLPTR (TBLPTRH.TBLPTRL), неявно адресуемого в командах TABLWT и TABLRD, используется в качестве адреса памяти программ. В указанных командах двухбайтный операнд для этих команд размещается в неявно адресуемом 16-разрядном регистре TBLATH.TBLATL. В операторах ассемблера, задающих пересылку между памятью программ и памятью данных, фигурируют три операнда: t, i, f, смысл которых ясен из приведенного выше описания формата команды. В операторах, задающих пересылку только между буфером данных TBLAT и памятью данных, фигурируют два операнда: t, f.

1.4. Система команд

При описании системы команд приняты следующие соглашения:

- в описаниях алгоритмов выполнения команд используются операторы языка Си;
- информация о влиянии команд на признаки результата представляется в последнем предложении спецификации команды списком флагов, которые меняются в соответствии с результатом операции; если команда не влияет ни на один признак результата, то список флагов не приводится;
- если время выполнения команды T_C не равно одному машинному циклу, то правило вычисления T_C приводится либо в конце спецификации, либо в преамбуле перед списком группы команд;
- результат двухместных операций *desc* размещается в WREG, если операнд d в операторе ассемблера равен 1; если он равен 0 или опущен, то *desc* размещается в f;
- конкатенация, т. е. состыковка, двух объектов x и y обозначается $x.y$, причем, x – старшая часть образуемого двоичного кода, а y – младшая.

Организация передачи управления

Любые переходы со сменой адреса в PC отменяют результат процесса загрузки следующей команды в конвейере команд. Тем самым длительность T_C команды, меняющей PC, нужно считать равной двум машинным циклам, а не одному. Все условные переходы реализуются как пропуск следующей команды. При этом вместо предвыбранной команды исполняется NOP, на фоне которого протекает выборка той команды, к которой ведет «пропуск».

Условные и безусловные переходы

`GOTO a` **Безусловный** переход по адресу *a*. TC = 2.

`BTFSC f, b` **Пропуск** следующей команды, если бит *b* регистра *f* равен 0. TC = 2.

`BTFSS f, b` Пропуск следующей команды, если бит *b* регистра *f* равен 1. TC = 2.

`CPFSEQ f` Пропуск следующей команды, если $f \neq$ WREG.

`CPFSGT f` Пропуск следующей команды, если $f >$ WREG.

`CPFSLT f` Пропуск следующей команды, если $f <$ WREG.

`DCFSNZ f` Пропуск следующей команды, если $(f = f - 1) \neq 0$.

`DECFSZ f` Пропуск следующей команды, если $(f = f - 1) = 0$.

`CFSNZ f` Пропуск следующей команды, если $(f = f + 1) \neq 0$.

INCFSZ*f* Пропуск следующей команды, если (*f* = *f* + 1) = 0.

TSTFSZ*f* Пропуск следующей команды, если *f* = 0.

Организация процедур, прерываний и прочее управление

CALL *a* Страничный вызов подпрограммы по адресу PC[15..13].[*a*], где *a* – 13 младших разрядов кода команды. Предварительно адрес возврата (PC + 1) сохраняется в стеке. После загрузки адреса перехода в PC его старший байт PCH загружается в PCLATH. TC = 2.

LCALL *k* Длинный вызов подпрограммы по адресу PCLATH.[*k*], где *k* – младший байт кода команды. Предварительно адрес возврата (PC + 1) сохраняется в стеке. Затем в PCL загружается *k*, а в PCH – PCLATH. TC = 2.

RETURN Возврат из подпрограммы. Из вершины стека извлекается и помещается в PC адрес возврата. Предварительно PCH загружается в PCLATH. TC = 2.

RETFIE Возврат из подпрограммы с разрешением прерываний. Используется для возврата из обработчика прерываний. PCLATH не изменяется. TC = 2.

RETLW *k* Возврат с константой в рабочем регистре. Сначала в PC загружается извлеченный из стека адрес возврата, а затем в WREG загружается константа *k*. PCLATH не изменяется. TC = 2.

SLEEP Вход в режим энергосбережения SLEEP: -TO = 1, -PD = WDT = 0, предделитель WDT = 0, остановка генератора тактов.

CLRWDT Очистить WDT: Постделитель WDT = Счетчик WDT = 0, -TO = -PD = 1.

NOP Нет операции. Формирует задержку в один машинный цикл.

Пересылка данных, установка и очистка бит

MOVLW *k* Переслать константу *k* в WREG.

MOVWF *f* Переслать WREG в *f*.

MOVFP *f*, *p* Переслать *f* в *p*.

MOVPF *f*, *p* Переслать *p* в *f*.

SWAPF *f* Переставить местами полубайты в *f*.

CLRF *f* Очистить *f*.

TABLRD *t*, *i*, *f* Табличное чтение из программной памяти: (*t* == 0? *f* = TBLATL: *f* = TBLATL), TBLAT = PMem[TBLPTR], TBLPTR += *i*. TC = (*f* == PCL)? 3:2.

TABLWT *t*,*i*,*f* Табличная запись в программную память: (*t* == 0? TBLATL = *f*: TBLATL = *f*, PMem[TBLPTR] = TBLAT, TBLPTR += *i*. TC = 2, если запись выполняется во внешнюю память программ, иначе запускается так называемая длинная запись, которая может быть прекращена только по прерыванию или сбросу.

TLRD *t*, *f* Чтение табличного буфера: (*t* == 0? *f* = TBLATL: *f* = TBLATL. Используется для пересылки в память данных одного из байтов слова, прочитанного командой TABLRD из программной памяти.

TLWT *t*, *f* Запись табличного буфера: (*t* == 0? TBLATL = *f*: TBLATL = *f*). Используется для загрузки в табличный буфер одного из байтов слова, которое затем будет записано в память программ командой TABLWT.

MOVLB *k* Переслать константу *k* в младший полубайт BSR.

MOVLR *k* Переслать константу *k* в старший полубайт BSR.

BCF *f*, *b* Очистить бит *b* в *f*.

BSF *f*, *b* Установить бит *b* в *f*.

Команды арифметической обработки

ADDWF *f*, *d* Сложение: *desc* = WREG + *f*. Флаги: OV,C,DC,Z.

ADDWFC *d*, *s* Сложение: *desc* = WREG + *f* + C. Флаги: OV,C,DC,Z.

ADDLW *k* Сложение: WREG с константой: WREG = WREG + *k*. Флаги: OV,C,DC,Z.

SUBWF *f*, *d* Сложение: *desc* = *f* - WREG. Флаги: OV,C,DC,Z.

SUBWFB *d*, *s* Сложение: *desc* = *f* – WREG – 1 + C. Флаги: OV,C,DC,Z.

SUBLW *k* Вычитание константы из WREG: WREG = WREG - *k*. Флаги: OV,C,DC,Z.

NEGW *f*, *s* Получение дополнения WREG: *f* = 0 - WREG, if(s = 0) WREG = *f*. Флаги: OV,C,DC,Z.

INCF *f*, *d* Инкремент: *desc* = *f* + 1. Флаги: OV,C,DC,Z.

DECF *f*, *d* Декремент: *desc* = *f* - 1. Флаги: OV,C,DC,Z.

`ICFSNZ`*f* Прибавить 1 к *f* и пропустить команду, если результат не равен 0.

`INCFSZ`*f* Прибавить 1 к *f* и пропустить команду, если результат равен 0.

`DCFSNZ`*f* Вычесть 1 из *f* и пропустить команду, если результат не равен 0.

`DECFSZ`*f* Вычесть 1 из *f* и пропустить команду, если результат равен 0.

`MULWF` *f* Перемножить беззнаковые WREG и *f*: PRODH.PRODL = WREG * *f*.

`MULLW` *k* Перемножить беззнаковые WREG и константу *k*: PRODH.PRODL = WREG * *k*.

`DAW` *f, s* Десятичная коррекция WREG: *f* = WREG,
(WREG[3..0] > 9 || DC == 1) *f*[3..0] += 6,
if (WREG[7..4] > 9 || C == 1) *f*[7..4] += 6,
if (*s* == 0) WREG = *f*. Флаги: C.

Команды логических операций

`COMF` *f, d* Инвертирование всех разрядов *f*: *desc* = 0xFF - *f*. Флаги: Z.\

`ANDWF` *f, d* Логическое И: *desc* = WREG & *f*. Флаги: Z.

`ANDLW` *k* Логическое И с константой: WREG &= *k*. Флаги: Z.

`IORWF` *f, d* Логическое ИЛИ: *desc* = WREG | *f*. Флаги: Z.

`IORLW` *k* Логическое ИЛИ с константой: WREG |= *k*. Флаги: Z.

`XORWF` *f, d* Исключающее ИЛИ: *desc* = WREG ^ *f*. Флаги: Z.

`XORLW` *k* Исключающее ИЛИ с константой: WREG^ = *k*. Флаги: Z.

`BTG` *f, b* Инвертировать бит *b* в *f*.

Команды сдвигов

`RLNCF` *f, d* Циклический сдвиг *f* влево: *desc* = *f*[6..0] *f*[7]. Справа вдвигается старший бит *f*.

`RLCF` *f, d* Циклический сдвиг *f* влево через перенос: C *desc* = *f* C. Справа вдвигается C, а старший бит выдвигается в C.

`RRNCF` *f, d* Циклический сдвиг *f* вправо: *desc* = *f*[0] *f*[7..1]. Слева вдвигается младший бит *f*.

`RRCF` *f, d* Циклический сдвиг *f* вправо через перенос: *desc* C = C *f*. Слева вдвигается C, а младший бит выдвигается в C.

2. Справочные данные по архитектуре микроконтроллеров AVR ATMega

2.1. Общая характеристика семейства ATMega

Существует несколько семейств МК AVR. Справочные данные, приводимые в этом разделе методических указаний, сформированы на основе описания семейств ATMega, приведенных в книгах [2,3]. Это старшие модели, процессорное ядро которых базируется на следующих основных архитектурных решениях:

- расширенная RISC архитектура;
- раздельное адресное пространство программ и данных с возможностью переносить данные из одного адресного пространства в другое;
- одноуровневая регистровая память данных, включающая в себя регистры общего и специального назначения;
- сочетание 8-разрядных операндов и 16-разрядных команд;
- двухуровневая конвейерная обработка, совмещающая выполнение текущей команды с выборкой следующей;
- набор из 130 команд, включающей в себя аппаратное умножение;
- возможность программировать программное ППЗУ как вне микропроцессорного устройства, так и без демонтажа микроконтроллера из устройства;
- широкий спектр встроенных периферийных устройств и средств сопряжения с внешними устройствами, включающий в себя: 53 линии ввода-вывода, 17 источников прерываний (два внешних и 15 внутренних), 3 таймера-счетчика (два 8-битных и один 16-битный), 3 входа захвата данных, широтно-импульсный модулятор (ШИМ) два 8-битных канала, 2 асинхронных приемопередатчика UART, 10-битный 10-канальный АЦП, синхронный последовательный порт SSP с SPI и I^2C;
- наличие режимов энергосбережения, сторожевого таймера и защиты доступа к памяти программ.

2.2. Программно-доступные компоненты и способы адресации

Основными программно-доступными компонентами AVR, задаваемыми в аргументах командных строк ассемблер-программы, являются:

- память программ (8К 16-битных ячеек flash-памяти для Atmega163);

- память данных (1К байт для Atmega163), включающая в себя регистры общего назначения и регистры ввода/вывода;
- регистр инструкций;
- программный счетчик;
- арифметико-логическое устройство;
- регистры общего назначения;
- регистр состояния;
- регистры ввода/вывода;
- стек.

Все адресное пространство флэш-памяти программ ($0000...$1FFF) разделено на два раздела: раздел программы начальной загрузки, размер которой может находиться в пределах от 256 до 2048 байт (...$1FFF), и раздел прикладной программы ($0000...). Оба раздела могут быть программно заблокированы от записи. Кроме того, в системе команд микроконтроллера предусмотрена специальная инструкция spm, осуществляющая запись данных в раздел прикладной программы. Эта команда может быть использована только в разделе программы начальной загрузки.

Весь массив 8К ячеек flash-памяти микроконтроллера ATMEGA163 разделен на 128 страниц по 64 слова. Размер раздела начальной загрузки может быть задан с помощью специальных fuse битов BOOTSZ: BOOTSZ1-BOOTSZ0 – 11, объем (слов) – 128, кол-во страниц – 2, адреса – $1F80 - $1FFF; 10 – 256, 4, $1F00 - $1FFF; 01 – 512, 8, $1E00 - $1FFF; 00 – 1024, 16, $1C00 - $1FFF.

В энергонезависимой flash-памяти микроконтроллеров могут присутствовать специализированные биты и байты, предназначенные для защиты программы пользователя и конфигурирования изделия. Эти ячейки памяти не отображаются в общем массиве памяти программ, имеют оригинальные имена и индивидуально программируются. Биты блокировки LB1 и LB2: 11 – защита отсутствует, 01 – запрет программирования Flash и EEPROM, 00 – запрет программирования и чтения Flash и EEPROM. Биты предохранители позволяют задавать некоторые конфигурационные особенности микроконтроллера. ATmega163, поддерживающий режим последовательного программирования, имеет бит SPIEN, который может запретить этот режим перезаписи кристалла. Кроме того, биты CKSEL[0..3] задают источник тактового сигнала микроконтроллера и время задержки старта микроконтроллера после сброса, а схемой контроля питания управляют биты BODEN и BODLEVEL.

В программной памяти находятся три области: сама память занимает адресное пространство $60…..$45F, 32 ячейки с адресами $0…1F – регистры общего назначения, 64 ячейки с адресами $2F…$5F – регистры ввода/вывода.

Ядро AVR имеет 32 регистра общего назначения. Каждому регистру соответствует дополнительно адрес в памяти данных, отображающий их в первых 32 ячейках пространства данных R0…R31 ($00…$1F). Шесть из 32 регистров (R26 …. R31), кроме обычной для прочих регистров функций, выполняют функцию 16-битных указателей адреса при косвенной адресации памяти данных и памяти программ. Эти три регистра косвенной адресации определяются как регистры X, Y и Z. При этом в четных регистрах хранятся младшие байты 16-битных переменных, а в нечетных – старшие.

В микроконтроллере ATmega163 предусмотрено 64 регистра ввода/вывода, которые размещены в пространстве ввода/вывода процессорного ядра по адресам от $00 по $3F. Для обращения к регистрам можно использовать специальные инструкции (IN) и вывода (OUT). При этом в инструкции необходимо указывать адрес регистра. Одновременно регистры ввода/вывода представлены также в адресном пространстве памяти данных и к ним можно адресоваться как к обычным ячейкам памяти с адресами от $20 до $5F, т. е. адрес получается простым добавлением $20 к непосредственному адресу регистра, в дальнейшем этот адрес приводится в круглых скобках. Регистры побитно адресуются специальными командами работы с битами: установки бита в регистре и очистки бита в регистре. Состояние каждого отдельного бита этих регистров может быть проверено командами переходов. Каждый регистр микроконтроллера имеет свое оригинальное имя, которое вместе с адресами регистром и описанием отдельных битов приводится ниже.

$3F ($5F): SREG	Регистр состояния, внутренняя структура <I,T,H,S,V,N,Z,C>, где C – флаг переноса, Z – флаг нуля, N – флаг отрицательного результата, V – флаг переполнения (дополнения до двух), S – флаг знака (S = N V), H – флаг полупереноса, T – бит сохранения копии, I – бит разрешения глобального прерывания (1 = 0 разрешено).
$3E ($5E) SPH	Указатель стека, старшая часть. Используются только три бита SP10-SP8.
$3D ($5D) SPL	Указатель стека, младшая часть – SP7-SP0.

$3B ($5B) GIMSK — Регистр маски прерываний. Биты: 7 [INT1] – внешнее прерывание INT1 разрешено (1 и бит I в регистре статуса SREG равен 1, то внешний вход запроса на прерывание INT1 становится активным), 6 [INT0] – внешнее прерывание INT0 разрешено.

$3A ($5A) GIFR — Регистр флагов прерываний.

$39 ($59) TIMSK — Регистр маски прерывания таймера/счетчика. Биты: 0 [TOIE0] – разрешение прерывания при переполнении счетчика (1 и установлен бит I в регистре SREG прерывание при переполнении разрешается, соответствующий вектор прерывания OVF имеет адрес $012, прерывание произойдет при установке бита TOV0 в регистре TIFR); 2 [TOIE1] – разрешение прерывания при переполнении таймера/счетчика 1; 3 [OCIE1B] – разрешение прерывания при равенстве содержимого счетчика и содержимого регистра сравнения OCR1B; 4 [OCIE1A] – разрешение прерывания при равенстве содержимого счетчика и содержимого регистра сравнения OCR1A; 5 [TICIE1] – разрешение прерывания при срабатывании входа захвата; 6 [TOIE2] – разрешение прерывания при переполнении таймера/счетчика 2, 7 [OCIE2] – разрешение прерывания выхода сравнения.

$38 ($58) TIFR — Регистр флагов прерывания таймера/счетчика. Биты: 0 [TOV0] – флаг переполнения таймера/счетчика 0 (устанавливается при переполнении таймера/счетчика 0 и сбрасывается аппаратно при выполнении соответствующего прерывания ($012) или программно при записи 0); 2 [TOV1] – флаг прерывания при переполнении таймера/счетчика 1; 3 [OCF1B] – флаг прерывания при равенстве содержимого счетчика и содержимого регистра сравнения OCR1B; 4 [OCF1A] – флаг прерывания при равенстве содержимого счетчика и содержимого регистра сравнения OCR1A; 5 [ICF1] – флаг прерывания при возникновении захвата, 6 [TOV2] – флаг прерывания по переполнению таймера/счетчика 2, 7 [OCF2] – флаг прерывания выхода сравнения.

$37 ($57) SPMCR Регистр управления SPM.

$36 ($56) TWCR Регистр управления передачи TWSI 2-проводного последовательного интерфейса. Биты: 0 [TWIE] – бит маски прерывания (1 и установлен бит I в SREG, то при установке флага TWINT происходит прерывание), 2 [TWEN] – флаг включения двухпроводного интерфейса, 3 [TWWC] – флаг ошибки передачи, 4 [TWSTO] – флаг остановки (если контроллер ведущий устанавливается для остановки передачи на линии, когда условие остановки выполнено аппаратно очищается), 5 [TWSTA] – флаг разрешения старта (1 если устройство объявляется ведущим (мастером) на двухпроводной шине, двухпроводной интерфейс аппаратно проверяет шину и разрешает установку бита, только если шина свободна), 6 [TWEA] – флаг разрешения подтверждения (1 – сигнал подтверждения ACK генерируется), 7 [TWINT] – флаг прерывания 2-проводного интерфейса (устанавливается аппаратно, когда 2-проводной последовательный интерфейс закончил выполнение текущего задания и находится в состоянии ожидания).

$35 ($55) MCUCR Регистр управления процессорным ядром. Биты: 1..0 [ISC01..ISC00] – активизация входа INT0 (00 – низкий уровень сигнала INT1 (INT0) генерирует запрос на прерывание, 01 – любое логическое изменение на INT1 (INT0) генерирует запрос на прерывание, 10 – задний фронт импульса на INT1 (INT0) генерирует запрос на прерывание, 11 – передний фронт сигнала на INT1 (INT0) генерирует запрос на прерывание), 3..2 [ISC11..ISC10] – активизация входа INT1 (аналогично ISC01..ISC00), 5..4 [SM1..SM0] – определяют режим энергосбережения (00 – холостого хода (Idle), 01 – шумоподавления (ADC Noise Reduction), 10 – ожидания (Power-down), 11 – экономии (Power Save)), 6 [SE] – разрешение режима энергосбережения.

$34 ($54) MCUSR Регистр состояния процессорного ядра позволяет определить источник сброса. Биты: 0 [PORF] – флаг сброса при включении питания (сброс – 0), 1 [EXTRF] – флаг внешнего сброса (сброс – 0), 2 [BORF] – флаг сброса при кратко-

временном провале питания (сброс – 0), 3 [WDRF] – флаг сторожевого таймера.

$33 ($53) TCCR0 Регистр управления таймером/счетчиком 0. Биты 2..0 [CS02..CS00] – задают коэффициент деления предделителя таймер/счетчика 0 и тип внешнего сигнала (000 – остановка таймера, 001 – CK, 010 – CK/8, 011 – CK/64, 100 – CK/256, 101 – CK/1024, 110 – внешний контакт T0 задний фронт, 111 – внешний контакт T0 передний фронт).

$32 ($52) TCNT0 Регистр данных таймера/счетчика 0.

$31 ($51) OSCCAL Регистр калибровки резонатора.

$30 ($50) SFIOR Регистр специальных функций ввода/вывода. Биты: 2 [PUD] – подключение индивидуальных подтягивающих резисторов для выводов портов; 3 [ACME] – подключение мультиплексора к аналоговому компаратору.

$2F ($4F) TCCR1A Регистр A управления таймера/счетчика 1.. Биты: 1..0 [PWM11..PWM10] – биты выбора режима широтно-импульсной модуляции (00 – работа таймера/счетчика 1 в ШИМ режиме запрещена, 01 – работа таймера/счетчика 1 в 8-разрядном ШИМ режиме, 10 – работа таймера/счетчика1 в 9-разрядном ШИМ режиме, 11 – работа таймера/счетчика 1 в 10-разрядном ШИМ режиме), 2 [FOC1B] воздействие на выход OC1B (1 – изменение на выводе OC1B относительно значений, установленных битами COM1B1 и COM1B0), 3 [FOC1A] – воздействие на выход OC1A (1 – изменение на выводе OC1A относительно значений, установленных COM1A1 и COM1A0), 5..4 [COM1B1..COM1B0] – заданит режима выхода B, 7..6 [COM1A1..COM1A0] – задание режима выхода A (00 – таймер/счетчик 1 отключен от вывода OC1X, 01 – переключение выходной линии OC1X, 10 – очистка выходной линии OC1X (на линии 0), 11 – установка выходной линии OC1X (1)).

$2E ($4E) TCCR1B Регистр B управления таймера/счетчика 1. Биты: 2..0 [CS12..CS10] – переключение входа счетчика/таймера 1 (000 – остановка таймера, 001 – CK, 010 –

CK/8, 011 – CK/64, 100 – CK/256, 101 – CK/1024, 110 – внешний контакт T1 задний фронт, 111 – внешний контакт T1 передний фронт); 6 [ICES1] – выбор фронта срабатывания на входе захвата 1 (0 – по падающему фронту на входе захвата 1С1Ю, 1 – по нарастающему); 7 [ICNC1] – установка режима подавления шума на входе захвата 1 (0 – функция подавления шума входного триггера захвата запрещена); 3 [CTC1] – очистка таймера/счетчика 1 по совпадению (1 – таймер/счетчик 1 при совпадении сбрасывается в состояние $0).

$2D ($4D) TCNT1H Регистр данных таймера/счетчика 1, старший байт.

$2C ($4C) TCNT1L Регистр данных таймера/счетчика 1, младший байт.

$2B ($4B) OCR1AH Регистр А выходного сравнения таймера/счетчика 1, старший байт.

$2A ($4A) OCR1AL Регистр А выходного сравнения таймера/счетчика 1, младший байт.

$29 ($49) OCR1BH Регистр В выходного сравнения таймера/счетчика 1, старший байт.

$28 ($48) OCR1BL Регистр В выходного сравнения таймера/счетчика 1, младший байт.

$27 ($47) ICR1H Регистры входного захвата таймера/счетчика 1, старший байт.

$26 ($46) ICR1L Регистры входного захвата таймера/счетчика 1, младший байт.

$25 ($45) TCCR2 Регистр управления таймера/счетчика 2.

$24 ($44) TCNT2 Регистр таймера/счетчика 2. Если данные записаны в регистр и выбран источник тактового сигнала, то счет продолжается с записанного значения.

$23 ($43) OCR2 Регистр выхода сравнения таймера/счетчика 2. Биты: 2..0 [CS22..CS20] – выбор тактового сигнала (000 – таймер/счетчик 2 остановлен, 001 – PCK2, 010 – PCK2/8, 011 – PCK2/32, 100 – PCK2/64, 101 – PCK2/128, 110 – PCK2/256, 111 – PCK2/1024), 3 [CTC2] – очистка

таймера/счетчика 2 по совпадению (1 – при совпадении сбрасывается в состояние $0), 5..4 [COM21..COM20] – режим выхода сравнения (00 – таймер/счетчик 1 отключен от вывода OC2, 01 – переключение выходной линии OC2, 10 – очистка выходной линии OC2 (на линии 0), 11 – установка выходной линии OC1X (на линии 1)), 6 [PWM2] – включение широтно-импульсного модулятора (1 – режим PWM), 7 [FOC2] – воздействие на выход 2 (1 – изменение состояния выхода сравнения OC2 относительно COM21 и COM20).

$22 ($42) ASSR Асинхронный регистр состояния. Биты: 0 [TCR2UB] – модернизация регистра контроля таймера/счетчика (1 – когда таймер/счетчик 2 работает асинхронно и регистр TCCR2 записывается), 1 [OCR2UB] – модернизация регистра выхода сравнения (1 – когда таймер/счетчик 2 работает асинхронно и OCR2 записывается, 0 – OCR2 готов к модификации); 2 [TCN2UB] – модификация таймера/счетчика 2 (1 – модифицируется, 0 – готов к модификации), 3 [AS2] – асинхронный режим таймера/счетчика 2 (0 – таймер/счетчик тактируется внутренним генератором СК, 1 – тактируется с контакта TOSC1).

$21 ($41) WDTCR Регистр управления сторожевым таймером. Биты 0..2 [WDP0….WDP2] – биты установки коэффициента предварительного деления сторожевого таймера и соответствующие им промежутки времени (000 – 16 K – 15 ms, 001 – 32 K – 30 ms, 010 – 64 K – 60 ms, 011 – 128 K – 0.12 s, 100 – 256 K – 0.24 s, 101 – 512 K – 0.49 s, 110 – 1.024 K – 0.97 s, 111 – 2,048 K – 1.9 s), 3 [WDE] – разрешение сторожевого таймера (1 – сторожевой таймер разрешен), 4 [WDTOE] – разрешение отключения сторожевого таймера (должен быть установлен в 1 при очистке бита WDE, иначе сторожевой таймер не будет запрещен).

$20 ($40) UBRRHI Регистры скорости передачи, старший байт.

$1F ($3F) EEARH Регистр адреса EEPROM, старший байт. Бит 0 [EEAR8] – старший бит адреса ячейки памяти.

$1E ($3E) EEARL	Регистр адреса EEPROM, младший байт. Биты 7..0 [EEAR8...EEAR0] – младший байт адреса ячейки памяти.
$1D ($3D) EEDR	Регистр данных EEPROM.
$1C ($3C) EECR	Регистр управления EEPROM. Биты: 0 [EERE] – разрешение чтения, 1 [EEWE] – разрешение записи (1 – запись разрешена, если бит EEMWE = 1), 2 [EEMWE] – управление разрешением записи, 3 [EERIE] – разрешение прерывания по готовности (1 и 1-бите регистра SREG равен 1 – разрешается прерывание по готовности EEPROM).
$1B ($3B) PORTA	Регистр данных, порт A
$1A ($3A) DDRA	Регистры направления, порт A. Любой бит = 1 – выход.
$19 ($39) PINA	Регистры входных контактов, порт A. При чтении любой бит = значение на контактах соответствующего порта.
$18 ($38) PORTB	Регистр данных, порт B
$17 ($37) DDRB	Регистры направления, порт B. Любой бит = 1 – выход.
$16 ($36) PINB	Регистры входных контактов, порт B. При чтении любой бит = значение на контактах соответствующего порта.
$15 ($35) PORTC	Регистр данных, порт C.
$14 ($34) DDRC	Регистры направления, порт C. Любой бит = 1 – выход.
$13 ($33) PINC	Регистры входных контактов, порт C. При чтении любой бит = значение на контактах соответствующего порта.
$12 ($32) PORTD	Регистр данных, порт D.
$11 ($31) DDRD	Регистры направления, порт D. Любой бит = – выход.
$10 ($30) PIND	Регистры входных контактов, порт D. При чтении любой бит = значение на контактах соответствующего порта.
$0F ($2F) SPDR	Регистр данных SPI.
$0E ($2E) SPSR	Регистр статуса SPI. Биты: 0 [SPI2X] – удвоение скорости передачи, 6 [WCOL] – флаг ошибки, 7 [SPIF] – флаг прерывания SPI.

$0D ($2D) SPCR — Регистр управления SPI (последовательный периферийный интерфейс). Биты: 1..0 [SPR1..SPR0] – выбор частоты синхронизации, SPI синхронизируется от тактового генератора процессорного ядра ведущего контроллера CK, с битом SPI2X регистра SPSR ([SPI2X, SPR1,SPR0]: 000 – CK/4, 001 – CK/16, 010 – CK/64, 011 – CK/128, 100 – CK/2, 101 – CK/8, 110 – CK/32, 111 – CK/64); 2 [CPHA] – фаза синхронизации (1 – по переднему фронту), 3 [CPOL] – полярность сигнала синхронизации (1 – SCK = 1 при отсутствии передачи), 4 [MSTR] – выбор ведущего/ведомого (1 – ведущий), 5 [DORD] – порядок передачи данных (1 – младший бит данных (LSB) передается первым), 6 [SPE] – разрешение работы SPI, 7 [SPIE] – бит маски прерывания SPI (1 и бит I глобального разрешения – прерывание разрешается).

$0C ($2C) UDR — Регистр данных UART (асинхронного последовательного интерфейса).

$0B ($2B) UCSRA — Регистр A управления и статуса UART. Биты: 0 [MPCM] – режим мультипроцессорного обмена (1 – ведомый контроллер принимает адресный байт), 1 [U2X] – удвоение скорости передачи, 3 [OR] – переполнение данных, 4 [FE] – ошибка кадра, 5 [UDRE] – регистр данных пуст, 6 [TXC] – передача завершена, 7 [RXC] – прием завершен.

$0A ($2A) UCSRB — Регистр B управления и статуса UART. Биты: 0 [TXB8] – передача 8-разрядных данных, 1 [RXB8] – прием 8-разрядных данных, 2 [CHR9] – режим 9-разрядных символов (1 – девятый разряд в RXB8 или TXB8), 3 [TXEN] – разрешение передачи данных, 4 [RXEN] – разрешение приемника, 5 [UDRIE] – разрешение прерывания по пустому регистру данных, 6 [TXCIE] – разрешение прерывания по завершению передачи, 7 [RXCIE] – разрешение прерывания по завершению приема.

$09 ($29) UBRR — Регистры скорости передачи (2400 бод: 1 МГц – 25, 2 МГц – 51, 4 МГц – 103, 8 МГц – 207; 4800 бод:

1 МГц – 12, 2 МГц – 25, 4 МГц – 51, 8 МГц – 103; 9600 бод: 2 МГц – 12, 4 МГц – 25, 8 МГц – 51; 14400 бод: 4 МГц – 16, 8 МГц – 34; 19200 бод: 4 МГц – 12, 8 МГц – 25; 28800 бод: 8 МГц – 16; 38400 бод: 8 МГц – 12).

$08 ($28) ACSR — Регистр управления аналоговым компаратором. Биты: 1..0 [ACIS1..ACIS0] – выбор режима прерывания (00 – прерывание по переключению выхода компаратора, 01 – зарезервировано, 10 – прерывание по падающему фронту на выходе компаратора, 11 – прерывание по нарастающему фронту на выходе компаратора); 2 [ACIC] – разрешение входа захвата входа таймера/счетчика (1 – захват разрешен); 3 [ACIE] – разрешение прерывания (1 – активируется прерывание ANA_COM с вектором $20); 4 [ACI] – флаг прерывания (устанавливается в 1 в случае формирования компаратором прерывания); 5 [ACO] – выход аналогового компаратора; 6 [ACBG] – выбор эталона (1 – фиксированное напряжение 1.22 В поступает на положительный вход аналогового компаратора, 0 – к положительному входу подключается контакт AIN0); 7 [ACD] – отключает аналоговый компаратор.

$07 ($27) ADMUX — Регистр управления мультиплексором. Биты: 4..0 [MUX4..MUX0] – выбора входа, коммутируемого на вход преобразователя (00000 – ADC0, 00001 – ADC1, 00010 – ADC2, 00011 – ADC3, 00100 – ADC4, 00101 – ADC5, 00110 – ADC6, 00111 – ADC7, 01000..11101 – резерв, 11110 – 1.22 В, 11111 – 0 В (AGND)), 5 [ADLAR] – воздействует на запись результата в регистры данных ADCL и ADCH (0 – можно использовать упрощенное 8-битное преобразование.), 7..6 [REFS1..0] – обеспечивают выбор эталонного напряжения на входе AREF аналого-цифрового преобразователя (00 – AREF, внутреннее напряжение Vref отключено; 01 – AVCC с внешним конденсатором на контакте AREF; 10 – резерв; 11 – внутренний источник 2.56 В с внешним конденсатором на AREF).

$06 ($26) ADCSR — Регистр управления аналого-цифровым преобразователем. Биты: 2..0 [ADPS2..ADPS0] – выбор коэффициента предварительного деления тактовой частоты микроконтроллера для получения необходимой тактовой частоты ADC (000 – без деления, 001 – 2, 010 – 4, 011 – 8, 100 – 16, 101 – 32, 110 – 64, 111 – 128); 3 [ADIE] – разрешение прерывания ADC (1 и I-бите SREG активируется прерывание с вектором $1C по завершению преобразования ADC); 4 [ADIF] – флаг прерывания ADC (устанавливается в 1 по завершению преобразования и обновления регистров данных); 5 [ADFR] – установка циклического режима; 6 [ADSC] – запуск преобразования ADC: 7 [ADEN] – разрешение работы ADC (1 – разрешено).

$05 ($25) ADCH — Регистр данных ADC, старший байт.

$04 ($24) ADCL — *Регистр данных ADC, младший байт. Биты* 7..6 [ADC1..ADC0] – младшая часть десятибитного результата.

$03 ($23) TWDR — Регистр данных. Хранит байт данных, для передачи или последний байт данных, полученный на 2-проводной последовательной шине.

$02 ($22) TWAR — Регистр адреса (ведомого). Бит 0 [TWGCE] – бит распознавания запроса. При установке бита выполняется операция чтения, при сбросе – операция записи.

$01 ($21) TWSR — *Регистр статуса. Доступен только для чтения. Биты* 7..3 [TWS] – отражают состояние логики последовательного интерфейса и 2-проводной последовательной шины.

$00 ($20) TWBR — Регистр задания скорости передачи TWSI 2-проводного последовательного интерфейса. Предназначен для задания скорости передачи данных. Частота синхронизации связана с содержимым регистра соотношением: частота SCL = тактовая частота ядра / (16 + 2TWBR).

2.3. Форматы команд и способы адресации

Процессорное ядро ATmega AVR реализует следующие основные форматы команд:

ccckkkkkkkddddd операции с 8-битными константами, где c..c – биты кода операции, k..k – биты константы, d..d – номер регистра;

cccccccccccddddd операции с одним регистром общего назначения;

cccccrrrrrddddd операции с двумя регистрами общего назначения, где r..r – регистр-приемник, d..d – регистр-источник;

ccccccccrrrrrbbb битовые операции с регистром общего назначения, где b..b – номер бита;

cccccdddddppppp операции с регистром ввода/вывода, где p..p – 5-разрядный адрес регистра ввода/вывода (с номером $P);

cccccccpppppbbb битовые операции с регистром ввода/вывода;

ccccccccccccrrrr kkkkkkkkkkkkkkkk операции с памятью данных, где r..r – номер регистра (приемника или источника), k..k – адрес ячейки памяти;

cccccccccccccck kkkkkkkkkkkkkkk операции с памятью данных, команды call и jmp, где k..k – адрес ячейки памяти;

Ccccddddqqqqqqq операции с памятью данных со смещением, где q..q – адрес прибавляемый к адресу содержащемуся в регистре.

В ядре МК реализованы следующие способы адресации:

Непосредственная адресация. В команде указан 8-битная константа и 5-битный номер регистра. В операторе ассемблера задается чаще всего имя регистра и числовая константа.

Прямая адресация. Операндами могут быть 8-битные переменные, хранящиеся в регистрах общего назначения, ячейках памяти данных и памяти программ или в регистрах ввода/вывода, а также отдельные биты этих регистров. Эти операнды указываются прямо в инструкции.

Косвенная адресация. Таким образом могут адресоваться ячейки памяти данных или памяти программ. К этому виду адресации относятся:

- адресация памяти данных с постинкрементом, когда операнд содержится в ячейке памяти данных, адрес операнда находится в одном из регистров косвенной адресации X, Y или Z, после выполнения операции регистр косвенной адресации (X, Y или Z) инкрементируется;

- адресация памяти данных с предде-крементом, когда операнд содержится в ячейке памяти данных, адрес операнда находится в одном из регистров косвенной адресации X, Y или Z, перед выполнения операции регистр косвенной адресации (X, Y или Z) декрементируется;
- адресация памяти данных со смещением, когда операнд содержится в ячейке памяти данных, адрес операнда вычисляется суммированием содержимого регистра Y или Z с 6 битами адреса, содержащимися в инструкции;
- адресация памяти программ, когда адрес константы в памяти программ определяется содержимым регистра Z или выполнение программы продолжается с адреса, записанного в регистре Z (программный счетчик загружается содержимым регистра Z).

Относительная адресация. Используется при обращении к памяти программ. В инструкциях перехода с помощью относительной адресации модифицируется содержимое программного счетчика.

2.4. Система команд

При описании системы команд приняты следующие соглашения:

- в описаниях алгоритмов выполнения команд используются операторы языка Си;
- информация о влиянии команд на признаки результата представляется в последнем предложении спецификации команды списком флагов, которые меняются в соответствии с результатом операции; если команда не влияет ни на один признак результата, то список флагов не приводится;
- если время выполнения команды T_C не равно одному машинному циклу, то правило вычисления T_C приводится либо в конце спецификации, либо в преамбуле перед списком группы команд;
- конкатенация, т. е. состыковка, двух объектов x и y обозначается $x:y$, причем, x – старшая часть образуемого двоичного кода, а y – младшая.

При описании системы команд использованы следующие обозначения:

- *Rd* - регистр-приемник результата ($0 \le d \le 31$),
- *Rd** - регистр-приемник результата с номером более 16 ($16 \le d \le 31$),
- *Rr* - регистр-источник ($0 \le r \le 31$),
- *Rdl*: регистры R24, R26, R28, R30 (для инструкций ADIW и SBIW),
- *P*- адрес регистра ввода/вывода,
- *P**- адрес побитно адресуемого регистра ввода/вывода ($00-$1F),
- *K* - символьная или численная константа (8 бит),

- *k* - адресная константа,
- *b* - номер бита в регистре (3 бита),
- *s* - номер бита в регистре статуса (3 бита),
- *X, Y, Z* - регистры косвенной адресации (*X* = R27 : R26, *Y* = R29 : R28; *Z* = R31 : R30).

Команды пересылки

Команды пересылки осуществляют перемещение данных между ячейками памяти и регистрами процессорного ядра. Один из операндов, участвующих в инструкции, является источником данных, второй – приемником. При пересылке из источника в приемник копия данных всегда остается в источнике. Одним из операндов в любой команде пересылки обычно является регистр общего назначения процессорного ядра. Вторым может быть любой регистр или ячейка памяти.

Время выполнения инструкций, работающих с регистрами, равно 1 такту. Инструкции, обращающиеся к ячейкам памяти данных, выполняются за 2 такта, а обращающиеся к ячейкам памяти программ, – за 3 такта.

`mov`	*Rd, Rr*	Пересылка между регистрами, *Rd* <- *Rr*.
`movw`	*Rd, Rr*	Пересылка между регистрами слова, *Rd + 1 : Rd* <- *Rr + 1 : Rr*.
`ldi`	*Rd*, K*	Загрузка непосредственного операнда, *Rd* <- *K*.
`ld`	*Rd, X*	Загрузка в регистр байта памяти, *Rd* <- *(X)*. TC = 2.
`ld`	*Rd, X+*	Загрузка в регистр байта памяти с постинкрементом, *Rd* <- *(X), X* <- *X + 1*. TC = 2.
`ld`	*Rd, -X*	Загрузка в регистр байта памяти с преддекрементом, *X* <- *X - 1, Rd* <- *(X)*. TC = 2.
`ld`	*Rd, Y*	Загрузка в регистр байта памяти, *Rd* <- *(Y)*. TC = 2.
`ld`	*Rd, Y+*	Загрузка в регистр байта памяти с постинкрементом, *Rd* <- *Y), Y* <- *Y + 1*. TC = 2.
`ld`	*Rd, -Y*	Загрузка в регистр байта памяти с преддекрементом, *Y* <- *Y - 1, Rd* <- *(Y)*. TC = 2.
`ldd`	*Rd, Y+q*	Загрузка в регистр байта памяти со смещением, *Rd* <- *(Y + q)*. TC = 2.
`ld`	*Rd, Z*	Загрузка в регистр байта памяти, *Rd* <- *(Z)*. TC = 2.
`ld`	*Rd, Z +*	Загрузка в регистр байта памяти с постинкрементом, *Rd* <- *(Z), Z* <- *Z + 1*. TC = 2.

ld	*Rd, - Z*	Загрузка в регистр байта памяти с преддекрементом, *Z <- Z - 1, Rd <- (Z)*. TC = 2.
ldd	*Rd, Z +q*	Загрузка в регистр байта памяти со смещением, *Rd <- (Z + q)*. TC = 2.
lds	*Rd <- (k)*	Загрузка в регистр байта памяти по его абсолютному адресу, *Rd <- (k)*. TC = 2.
st	*X, Rr*	Сохранение регистра в памяти, *(X) <- Rr*. TC = 2.
st	*X+, Rr*	Сохранение регистра в памяти с постинкрементом, *(X) <- Rr, X <- X + 1*. TC = 2.
st	*- X, Rr*	Сохранение регистра в памяти с преддекрементом, *X <- X - 1, (X) <- Rr*. TC = 2.
st	*Y, Rr*	Сохранение регистра в памяти, *(Y) <- Rr*. TC = 2.
st	*Y+, Rr*	Сохранение регистра в памяти с постинкрементом, *(Y) <- Rr, Y <- Y + 1*. TC = 2.
st	*-Y, Rr*	Сохранение регистра в памяти с преддекрементом, *Y <- Y - 1, (Y) <- Rr*. TC = 2.
std	*Y+q,Rr*	Сохранение регистра в памяти со смещением, *(Y + q) <- Rr*. TC = 2.
st	*Z, Rr*	Сохранение регистра в памяти, *(Z) <- Rr*. TC = 2.
st	*Z+, Rr*	Сохранение регистра в памяти с постинкрементом, *(Z) <- Rr, Z <- Z + 1*. TC = 2.
st	*-Z, Rr*	Сохранение регистра в памяти с преддекрементом, *Z <- Z - 1, (Z) <- Rr*. TC = 2.
std	*Z+q,Rr*	Сохранение регистра в памяти со смещением, *(Z + q) <- Rr*. TC = 2.
sts	*k, Rr*	Сохранение регистра в памяти по абсолютному адресу, *(k) <- Rr*. TC = 2.
lpm		Загрузка памяти программ, *R0 <- (Z)*. TC = 3.
lpm	*Rd, Z*	Загрузка в регистр памяти программ, *Rd <- (Z)*. TC = 3.
lpm	*Rd, Z+*	Загрузка в регистр памяти программ с постинкрементом, *Rd <- (Z), Z = Z + 1*. TC = 3.
spm		Сохранение памяти программ, *(Z) <- R1 : R0*.
in	*Rd, P*	Чтение данных из порта, *Rd <- P*. TC = 1.
out	*P, Rr*	Вывод данных в порта, *P <- Rr*. TC = 1.

`push`	*Rr*	Сохранение регистра в стеке, *STACK <- Rr; SP <- SP - 1.* TC = 2.
`pop`	*Rd*	Извлечение данных из стека в регистр, *SP <- SP + 1, Rd <- STACK.* TC = 2.

Арифметические и логические команды

В группу арифметических команд входят команды, выполняющие сложение, вычитание, декремент и инкремент данных, логическое умножение, логическое сложение, операцию ИСКЛЮЧАЮЩЕЕ ИЛИ, инверсию переменной. Обычно к этой группе относят также инструкции сравнения данных. В микроконтроллере ATmega163 реализованы также функции арифметического умножения целых чисел и дробных чисел, без знака и со знаком.

Все инструкции этой группы, как правило, приводят к изменению состояния флагов регистра состояния в соответствии с результатами, выполняемой операции.

`add`	*Rd, Rr*	Сложение регистров, *Rd <- Rd + Rr.* Флаги: Z,C,N,V,H. TC = 1.
`adc`	*Rd, Rr*	Сложение регистров с флагом переноса, *Rd <- Rd + Rr + C.* Флаги: Z,C,N,V,H. TC = 1.
`adiw`	*Rdl, K*	Сложение слова в регистрах с непосредственным операндом, *Rdh : Rd <- Rdh : Rdl + K.* Флаги: Z,C,N,V,S. TC = 2.
`sub`	*Rd, Rr*	Вычитание регистров, *Rd <- Rd – Rr.* Флаги: Z,C,N,V,H. TC = 1.
`subi`	*Rd*, K*	Вычитание константы, *Rd <- Rd – K.* Флаги: Z,C,N,V,H. TC = 1.
`sbc`	*Rd, Rr*	Вычитание регистров с флагом переноса, *Rd <- Rd – Rr – C.* Флаги: Z,C,N,V,H. TC = 1.
`sbci`	*Rd*, K*	Вычитание константы с флагом переноса, *Rd <- Rd – K – C.* Флаги: Z,C,N,V,H. TC = 1.
`sbiw`	*Rdl, K*	Вычитание 16-разрядной константы, *Rdh : Rd <- Rdh : Rdl – K.* Флаги: Z,C,N,V,S. TC = 2.
`and`	*Rd, Rr*	Логическое И, *Rd <- Rd Rr.* Флаги: Z,N,V. TC = 1.
`andi`	*Rd*, K*	Логическое И регистра и константы, *Rd <- Rd K.* Флаги: Z,N,V. TC = 1.
`or`	*Rd, Rr*	Логическое ИЛИ, *Rd <- Rd v Rr.* Флаги: Z,N,V. TC = 1.

`ori`	*Rd*, K*	Логическое ИЛИ регистра и константы, *Rd <- Rd v K.* Флаги: Z,N,V. TC = 1.
`eor`	*Rd, Rr*	Исключающее ИЛИ, *Rd <- Rd Rr.* Флаги: Z,N,V. TC = 1.
`com`	*Rd*	Дополнения до единицы, *Rd <- $FF – Rd.* Флаги: Z,C,N,V. TC = 1.
`neg`	*Rd*	Дополнения до двух, *Rd <- $00 – Rd.* Флаги: Z,C,N,V,H. TC = 1.
`sbr`	*Rd*, K*	Установка бита в регистре, *Rd <- Rd v K.* Флаги: Z,N,V. TC = 1.
`ser`	*Rd*	Установка всех битов в регистре, *Rd <- $FF.* TC = 1.
`cbr`	*Rd*, K*	Сброс бита в регистре, *Rd <- Rd ($FF – K).* Флаги: Z,N,V. TC = 1.
`clr`	*Rd*	Очистка регистра, *Rd <- $00.* Флаги: Z,N,V. TC = 1.
`inc`	*Rd*	Инкремент, *Rd <- Rd + 1.* Флаги: Z,N,V. TC = 1.
`dec`	*Rd*	Декремент, *Rd <- Rd - 1.* Флаги: Z,N,V. TC = 1.
`tst`	*Rd*	Проверка на ноль или минус, *Rd <- Rd Rd.* Флаги: Z,N,V. TC = 1.
`cp`	*Rd, Rr*	Сравнение, *Rd – Rr.* Флаги: Z,C,N,V,H. TC = 1.
`cpc`	*Rd, Rr*	Сравнение с учетом флага переноса, *Rd – Rr - C.* Флаги: Z,C,N,V,H. TC = 1.
`cpi`	*Rd*, K*	Сравнение с константой, *Rd – K.* Флаги: Z,C,N,V,H. TC = 1.
`mul`	*Rd, Rr*	Умножение беззнаковое, *R1 : R0 <- Rd x Rr.* Флаги: Z,C. TC = 2.
`muls`	*Rd, Rr*	Умножение знаковое, *R1 : R0 <- Rd x Rr.* Флаги: Z,C. TC = 2.
`mulsu`	*Rd, Rr*	Знаковое умножение беззнаковых операндов, *R1 : R0 <- Rd x Rr.* Флаги: Z,C. TC = 2.
`fmul`	*Rd, Rr*	Беззнаковое умножение дробных операндов, *R1 : R0 <- (Rd x Rr) << 1.* Флаги: Z,C. TC = 2.
`fmuls`	*Rd, Rr*	Знаковое умножение дробных операндов, *R1 : R0 <- (Rd x Rr) << 1.* Флаги: Z,C. TC = 2.
`fmulsu`	*Rd, Rr*	Знаковое умножение беззнаковых дробных операндов, *R1 : R0 <- (Rd x Rr) << 1.* Флаги: Z,C. TC = 2.

Битовые команды

Битовые команды позволяют обращаться непосредственно к отдельным битам регистров процессорного ядра и выполнять с выбранными битами простейшие операции: пересылки, установки и сброса. Операндами команд могут быть как биты регистров общего назначения, так и биты регистров ввода/вывода. Битовые команды могут влиять на отдельные флаги регистра признаков.

`sbi` *P*,b* Установка бита в регистре ввода/вывода, *I/O(P,b) <- 1.* TC = 2.

`cbi` *P*,b* Сброс бита в регистре ввода/вывода, *I/O(P,b) <- 0.* TC = 2.

`lsl` *Rd* Логический сдвиг влево, *Rd(n + 1) <- Rd(n), Rd(0) <- 0.* Флаги: Z,C,N,V. TC = 1.

`lsr` *Rd* Логический сдвиг вправо, *Rd(n) <- Rd(n + 1), Rd(7) <- 0.* Флаги: Z,C,N,V. TC = 1.

`rol` *Rd* Циклический сдвиг влево через флаг переноса, *Rd(0) <- C, Rd(n + 1) <- Rd(n), C <- Rd(7).* Флаги: Z,C,N,V. TC = 1.

`ror` *Rd* Циклический сдвиг вправо через флаг переноса, *Rd(7) <- C, Rd(n) <- Rd(n + 1), C <- Rd(0).* Флаги: Z,C,N,V. TC = 1.

`asr` *Rd* Арифметический сдвиг вправо, *Rd(n) <- Rd(n + 1), n = 0..6.* Флаги: Z,C,N,V. TC = 1.

`swap` *Rd* Переставить местами полубайты, *Rd(3..0) <- Rd(7..4), Rd(7..4) <- Rd(3..0).* TC = 1.

`bset` *s* Установка флага, *SREG(s) <- 1 SREG(s).* TC = 1.

`bclr` *s* Сброс флага, *SREG(s) <- 0 SREG(s).* TC = 1.

`bld` *Rd, b* Загрузка бита в регистр из флага сохранения копии , *Rd(b) <- T.* TC = 1.

`bst` *Rr, b* Установка флага сохранения копии из регистра, *T <- Rr(b).* Флаги: T. TC = 1.

`sec` Установка флага переноса, *C <- 1.* Флаги: C. TC = 1. .

`clc` Сброс флага переноса, *C <- 0.* Флаги: C. TC = 1.

`sen` Установка флага N, *N <- 1.* Флаги: N. TC = 1.

`cln` Сброс флага N, *N <- 0.* Флаги: N. TC = 1.

`sez` Установка флага нуля, *Z <- 1.* Флаги: Z. TC = 1.

`clz` Сброс флага нуля, *Z <- 0.* Флаги: Z. TC = 1.

`sei` Разрешение глобального прерывания, *I <- 1.* Флаги: I. TC = 1.

`cli`	Запрещение глобального прерывания, *I <- 0.* . Флаги: I. TC = 1.
`ses`	Установка флага знака, *S <- 1.* Флаги: S. TC = 1.
`cls`	Сброс флага знака, *S <- 0.* Флаги: S. TC = 1.
`sev`	Установка флага переполнения, *V <- 1.* Флаги: V. TC = 1.
`clv`	Сброс флага знака переполнения, *V <- 0.* Флаги: V. TC = 1.
`set`	Установка флага сохранения копии в регистре состояния, *T <- 1.* Флаги: T. TC = 1.
`clt`	Сброс флага сохранения копии в регистре состояния, *T <- 0.* Флаги: T. TC = 1.
`she`	Установка флага полупереноса, *H <- 1.* Флаги: H. TC = 1.
`clh`	Сброс флага полупереноса, *H <- 0.* Флаги: H. TC = 1.

Инструкции ветвления

Инструкции ветвления изменяют содержимое программного счетчика. Они используются при организации переходов и подпрограмм. Инструкции условных переходов и условных вызовов подпрограмм в процессе исполнения проверяют выполнение различных условий. Если условие не выполняется, программный счетчик просто инкрементируется.

Время выполнения команд ветвления зависит от результата проверки. Оно, для различных инструкций, может меняться в пределах от одного до четырех тактов.

`rjmp` *k*	Относительный переход, *PC <- PC + k + 1.* TC = 2.
`ijmp`	Переход по адресу Z, *PC <- Z.* TC = 2.
`jmp` *k*	Переход, *PC <- k.* TC = 3.
`rcall` *k*	Относительны вызов процедуры, *PC <- PC + k + 1.* TC = 3.
`call` *k*	Вызов процедуры, *PC <- k.* TC = 4.
`icall`	Вызов процедуры по абсолютному адресу, *PC <- Z.* TC = 3.
`ret`	Возврат из процедуры, *PC <- STACK.* TC = 4.
`reti`	Возврат из прерывания, *PC <- STACK.* Флаги: I. TC = 4.
`cpse` *Rd, Rr*	Сравнение, пропуск, если равно, *if (Rd = Rr) then PC <- PC + 2 o 3.* TC = 1/2/3.
`sbrc` *Rr, b*	Пропуск, если бит в регистре очищен, *if (Rr(b) = 0) then PC <- PC + 2 or 3.* TC = 1/2.
`sbrs` *Rr, b*	Пропуск, если бит в регистре установлен, *if (Rr(b) = 1) then PC <- PC + 2 or 3.* TC = 1/2.

`sbic` *P*, b* Пропуск, если бит в регистре ввода/вывода очищен, *if (P(b) = 0) then PC <- PC + 2 or 3*. TC = 1/2.

`sbis` *P*, b* Пропуск, если бит в регистре ввода/вывода установлен, *if + (P(b) = 1) then PC <- PC + 2 or 3*. TC = 1/2.

`brbs` *s, k* Перейти, если флаг в регистре статуса установлен, *if (SREG(s) = 1) then PC <- PC + k + 1*. TC = 1/2.

`brbc` *s, k* Перейти, если флаг в регистре статуса очищен, *if (SREG(s) = 0) then PC <- PC + k + 1*. TC = 1/2.

`breq` *k* Перейти, если равно, *if (Z = 1) then PC <- PC + k + 1*. TC = 1/2.

`brcs` *k* Перейти, если был перенос, *if (C = 1) then PC <- PC + k + 1*. TC = 1/2.

`brne` *k* Перейти, если не равно, *if (Z = 0) then PC <- PC + k + 1*. TC = 1/2.

`brcc` *k* Перейти, если не было переноса, *if (C = 1) then PC <- PC + k + 1*. TC = 1/2.

`brsh` *k* Перейти, если больше или равно, *if (C = 0) then PC <- PC + k + 1*. TC = 1/2.

`brlo` *k* Перейти, если меньше, *if (C = 1) then PC <- PC + k + 1*. TC = 1/2.

`brmi` *k* Перейти, если минус, *if (N = 1) then PC <- PC + k + 1*. TC = 1/2.

`brpl` *k* Перейти, если плюс, *if (N = 0) then PC <- PC + k + 1*. TC = 1/2.

`brge` *k* Перейти, если больше или равно для знаковых операндов, *if (N ⊕ V = 0) then PC <- PC + k + 1*. TC = 1/2.

`brlt` *k* Перейти, если меньше нуля для знаковых операндов, *if (N ⊕ V = 1) then PC <- PC + k + 1*. TC = 1/2.

`brhs` *k* Перейти, если был полуперенос, *if (H = 1) then PC <- PC + k + 1*. TC = 1/2.

`brhc` *k* Перейти, если не было полупереноса, *if (H = 0) then PC <- PC + k + 1*. TC = 1/2.

`brts` *k* Перейти, если флаг сохранения копии установлен, *if (T = 1) then PC <- PC + k + 1*. TC = 1/2.

`brtc`	*k*	Перейти, если флаг сохранения копии сброшен, *if (T = 0) then PC <- PC + k + 1*. TC = 1/2.
`brvs`	*k*	Перейти, если переполнение, *if (V = 1) then PC <- PC + k + 1*. TC = 1/2.
`brvc`	*k*	Перейти, если не было переполнения, *if (V = 0) then PC <- PC + k + 1*. TC = 1/2.
`brie`	*k*	Перейти, если прерывания разрешены, *if (I = 1) then PC <- PC + k + 1*. TC = 1/2.
`brid`	*k*	Перейти, если прерывания запрещены, *if (I = 0) then PC <- PC + k + 1*. TC = 1/2.

Инструкции управления

Инструкции управления используются для управления вычислениями и режимами работы процессорного ядра. Пустая операция nop заполняет пробелы между инструкциями в программной памяти или используется в качестве выдержки. Команда sleep переводит процессорное ядро в режим пониженного энергопотребления. Инструкция wdr сбрасывает сторожевой таймер.

`nop`	Пустая операция. TC = 1.
`sleep`	Перевод процессорного ядра в режим пониженного энергопотребления. TC = 3.
`wdr`	Сброс сторожевого таймера. TC = 1.

Библиографический список

1. Микроконтроллеры. Выпуск 1. – М.: ДОДЭКА, 1998. – 384 с.
2. Предко, М. Справочник по PIC-микроконтроллерам: пер. с англ. / М. Предко – М.: ДМК Пресс, 2004; ООО «Издательский дом «Додэка-XXI», 2004. – 512 с.
3. Предко, М. Руководство по микроконтроллерам. Том II. / М. Предко – М.: Постмаркет, 2001. – 488 с.
4. Евстифеев, А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «ATMEL» / А. В. Евстифеев – М.: Издательский дом «Додэка-XXI», 2004. – 560 с.
5. Бродин, В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М.: ЭКОМ, 2002. – 400 с.

Учебное издание

Справочные данные по программированию микроконтроллеров

Составители: НЕГОДА Виктор Николаевич, ЛЫЛОВА Анна Вячеславовна, РАТАНОВА Ольга Валентиновна

Редактор Н. А. Евдокимова

Подписано в печать 30.09.2006. Формат 60x84 1/16. Бумага офсетная.

Печать трафаретная. Усл. печ. л. 2,79. Уч.-изд.л. 2,00.Тираж 100 экз.

Заказ №1209

Ульяновский государственный технический университет,

432027, г.Ульяновск, ул. Северный Венец, 32.

Типография УлГТУ, 432027, г.Ульяновск, ул. Сев. Венец, 32.