«Қостанай жоғары политехникалық колледжі» КМҚК

КГКП «Костанайский политехнический высший колледж»

Учебно-методический комплекс

КВ 04 «ПРОГРАММИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ НА БАЗЕ МИКРОКОНТРОЛЛЕРОВ»

СОДЕРЖАНИЕ

Лекция 1.

Тема: Общее устройство МК, организация памяти

Микроконтроллер – это такая микросхема, которая представляет собой мини-компьютер, предназначенный для выполнения различных функций. Данная микросхема работает в соответствии с заложенной в нее программой, которую создает программист. Микроконтроллер может в себе содержать различное количество так называемых периферийных модулей, которые определяю его возможности, а также стоимость. К периферии микроконтроллера относятся, например: АЦП (аналого-цифровой преобразователь), различные таймеры, аналоговый компаратор, UART (по-простому говоря COM-порт), USB, CANи т.д. Но, как правило, любой микроконтроллер содержит следующие основные узлы:

- Арифметико-логическое устройство (АЛУ или ALU);
- Оперативная память (ОЗУ);
- Постоянная память (ПЗУ);
- Генератор тактовой частоты;
- Порты ввода/вывода;
- Таймеры;

Сердцем микроконтроллера является арифметико-логическое устройство (АЛУ). АЛУ производит все арифметические и логические операции с двоичными данными. Бывают АЛУ различной разрядности: 8-, 16- или 32-разрядные. Например, если АЛУ 8-разрядное, то оно может провести операцию над двумя восьмиразрядными числами и выдать восьмиразрядный результат операции.

К арифметическим операциям относятся: сложение, вычитание, сравнение и т.д.

К логическим операциям относятся: операция умножения «И», сложения «ИЛИ», отрицания «НЕ», «исключающее ИЛИ», сдвиг вправо, сдвиг влево и т.д. Есть также операции, которые не относятся ни к логическим, ни к арифметическим, например сброс в «0» или установка в «1».

Как было сказано выше, АЛУ производит операции над числами и возвращает результат операции в виде числа. Данные числа помещаются в регистры общего назначения – своеобразную временную память. У каждого микроконтроллера количество регистров может быть разным. На структурной схеме приведен пример контроллера, у которого 32 регистра общего назначения.

Однако, для нормальной работы микроконтроллера регистров общего назначения недостаточно, т.к., например, 32 байта – очень маленький объем памяти. Для того, чтобы можно было хранить больше информации, используется оперативно-запоминающее устройство (ОЗУ). Регистры общего

назначения содержат данные, с которыми АЛУ работает в данный момент, а ОЗУ – остальные.

Команды, а точнее последовательность команд, которые выполняет АЛУ, хранятся в постоянно-запоминающем устройстве (ПЗУ). Обычно это Flash-память. Данная последовательность команд является ничем иным, как программой микроконтроллера, которую создает программист. Все команды находятся в ПЗУ по определенным адресам.

Для того, чтобы достать какую-то команду из ПЗУ, необходимо обратиться к ее адресу, чем занимается программный счетчик или счетчик команд.

Данные из ПЗУ попадают в регистр команд. АЛУ постоянно «смотрит» содержимое регистра команд и если в нем появляется команда, то АЛУ сразу же начинает ее выполнять.

Все эти устройства микроконтроллера были бы бесполезны без портов ввода-вывода, с помощью которых микроконтроллер взаимодействует с внешним миром. Порты ввода-вывода можно настраивать таким образом, чтобы они работали как в качестве входов, так и в качестве выходов. Управления портами осуществляется через специальные регистры. По умолчанию все порты микроконтроллера настроены на выход.

Большинство современных микроконтроллеров имеют Гарвардскую архитектуру и содержат 3 вида памяти:

1. память программ FLASH;
2. оперативная память (ОЗУ) SRAM (Static RAM);
3. энергонезависимая память данных EEPROM.

Адресные пространства указанных видов памяти, как правило, разделены. Способы адресации и доступа к этим областям памяти также различны. Такая структура позволяет центральному процессору работать одновременно как с памятью программ, так и с памятью данных, что существенно увеличивает производительность. Каждая из областей памяти данных (SRAM и EEPROM) также расположена в своем адресном пространстве.

Память программ

Память программ представляет собой электрически стираемое ППЗУ (FLASH) и может поддерживать команды с разрядностью больше 8 бит. В некоторых микроконтроллерах память программ разделена на 2 секции:

- секцию загрузчика (Boot Program);
- секцию прикладных программ (Application Program).

Память программ чаще всего является электрически перепрограммиру-емой, количество циклов перезаписи превышает 10 тысяч.

Большинство микроконтроллеров поддерживают внутрисхемное программирование, т. е. загрузку программы в микроконтроллер можно осуществлять после монтажа на плату посредством специального разъема программирования. Для адресации памяти программ используется счетчик команд (Program Counter – PC).

В памяти программ также находится вектор сброса — в момент подачи питания микроконтроллер начинает выполнение программы с этого адреса, и здесь размещается команда перехода к началу исполняемой программы. Кроме того, память программ содержит таблицу векторов прерываний. При возникновении прерывания после сохранения в стеке текущего значения счетчика команд происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний. Положение вектора сброса и таблицы векторов прерываний может быть перенесено из секции прикладных программ в секцию загрузчика.

В некоторых случаях память программ может использоваться не только для хранения кода программы, но и для хранения различных констант.

Оперативная память

Оперативная память, как правило, содержит 3 области:

1. регистры общего назначения;
2. служебные регистры;
3. память для хранения данных.

Регистры общего назначения (РОН) находятся в непосредственной близости к АЛУ. Однако в микроконтроллерах некоторых фирм (в частности, PIC фирмы Microchip) имеется только один рабочий регистр, играющий роль одного из операндов в командах.

Применение набора регистров общего назначения в сочетании с конвейерной обработкой позволяет АЛУ выполнять одну операцию (извлечение операндов из набора регистров, выполнение команды и запись результата обратно в регистр) за один такт.

Служебные регистры имеют свои имя, адрес и назначение. Они предназначены для конфигурации и обслуживания периферийных узлов микроконтроллера. Краткая характеристика служебных регистров должна быть приведена в руководстве по использованию микроконтроллера (Data Sheet).

Среди служебных регистров есть, как правило, один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния. Он содержит набор флагов, показывающих текущее состояние

микроконтроллера. Большинство флагов автоматически устанавливаются в «1» или сбрасываются в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи. Эта информация анализируется при выполнении условных переходов. При возникновении прерываний содержимое регистра состояния необходимо сохранять программно (чаще всего это является «заботой» компилятора).

Остальная часть оперативной памяти предназначена для хранения пользовательских данных.

Энергонезависимая память данных

Энергонезависимая память данных (EEPROM) организована таким образом, что содержимое каждого байта отдельно может быть считано или записано. Количество циклов перезаписи энергонезависимой памяти превышает 100 тысяч. Энергонезависимая память предназначена для хранения настроек и конфигурации программы, то есть тех данных, которые должны сохраняться при пропадании питания.

Чтение и запись данных в EEPROM, как правило, осуществляется посредством использования соответствующих регистров из области служебных регистров SRAM. Как правило, это:

- регистр адреса при обращении к EEPROM;
- регистр данных, считанных/записанных в EEPROM;
- регистр управления чтением-записью EEPROM.

Лекция 2.

Тема: Система команд AVR

Микроконтроллеры AVR имеют систему сокращенного набора команд RISC, хотя целиком и не полностью попадают под это определение.

Система RISC подразумевает полную симметрию между ресурсами памяти разного типа. Это, в частности, позволяет обращаться к регистрам, портам и памяти данных одними и теми же командами, что и обуславливает их небольшое количество. Однако несмотря на то, что адресное пространство памяти AVR действительно непрерывно, всё же три разных его области используются только для своих специфических целей. РОН – преимущественно для математических операций, РВВ – для управления процессором, ОЗУ – только как хранилище информации. В связи с этим существуют группы команд как для работы с каждым видом памяти в отдельности, так и для пересылки данных из одной области памяти в другую. Поэтому и количество команд AVR достаточно велико. В фирменной документации, где много говорится про ортогональность ядра, в первую очередь имеется в виду полная равноправность именно РОН.

Система команд линейки ATtiny является подмножеством системы команд старшего семейства ATmega. В ряде старых моделях ATtiny могут отсутствовать некоторые аппаратные узлы (индексные регистры X, Y, программный стек, память ОЗУ и др.) и, соответственно, отсутствовать связанные с ними команды. Система же команд ATtiny более позднего времени выпуска практически аналогична семейству ATmega. Главное отличие обеих семейств – отсутствие встроенного умножителя у ATtiny (отсутствие группы команд умножения).

Разные модели ATtiny могут иметь (90…120) команд. ATmega поддерживают (130…135) инструкций. Так заявлено в спецификациях Atmel. Но фактическое число, на самом деле, значительно меньше.

Это связано с тем, что в ассемблере AVR встроен ряд макроопределений, эквивалентных реальным командам, но имеющих иной символический вид. Так, например, у команды ori Rd,K существует команда двойник sbr Rd,K, которая выполняет тоже действие (Rd = Rd OR K) и имеет такой же код операции. Аналогичные псевдокоманды существуют и для разных случаев применения bset s, bclr s, brbs s,k, brbc s,k, и мн. др.

Способы адресации

Большинство команд ассемблера используют различные ячейки памяти микроконтроллера и, соответственно, содержат кроме кода операции (КОП) также их адреса. В зависимости от того в каком виде в команде хранится адрес различают два способа адресации: прямую и косвенную. В первом случае адрес ячейки задан явно, а во-втором он находится в одном из

регистров-указателей (у AVR это 16-разрядные регистры X,Y,Z). У микропроцессоров различного типа, оба способа адресации могут иметь множество вариаций. Ниже приведены характерные только для микроконтроллеров AVR.

Разновидности прямой адресации

Прямая адресация регистра общего назначения

В команде присутствует адрес регистра приемника либо источника результата. Примерами команд могут служить inc Rd, dec Rd, lsl Rd, lsr Rd и т.д. Адресацию, где в команде кроме адреса регистра находится еще и константа (ldi Rd,K, ori Rd,K и т.д.), называют также непосредственной, а в случае сохранения/восстановления данных в стеке (push Rr и pop Rd) – стековой. Адрес РОН находится в пределах 0…31 (в командах с непосредственной адресацией 0…15).

Прямая адресация двух регистров общего назначения

Команды данного типа содержат адреса двух РОН, один из которых является источником, а другой приемником результата в арифметических операциях, а также операциях пересылки. Примеры команд: mov Rd, Rr, add Rd, Rr, sub Rd, Rr, and Rd,Rr и т.д. Адреса обоих регистров лежат в пределах 0…31, но в некоторых командах умножения могут использоваться только РОНы 16…31 (muls Rd,Rr и fmuls Rd,Rr) или 16…23 (mulsu Rd,Rr и fmulsu Rd,Rr).

Прямая адресация регистра ввода-вывода

Прямую адресацию регистра ввода-вывода у AVR используют команды двух типов. Это копирование РВВ в РОН in Rd,P и пересылка в противоположном направлении out P,Rr. В обоих случаях могут быть использованы любые регистры общего назначения 0…31 и регистры ввода-вывода 0…63.

Прямая адресация ОЗУ

Прямая адресация ОЗУ встречается в командах lds Rd,k и sts k,Rr. Первая инструкция пересылает байты из SRAM микроконтроллера в один из РОНов, вторая копирует содержимое РОНа в ячейку SRAM. В обеих командах под поле адреса ячейки памяти отводится 16 битов, а значит, имеется возможность напрямую обращаться к любому адресу SRAM из диапазона 0…65535. Инструкции работают со всеми РОНами 0…31 и имеют размер в 2 слова (4 байта).

Разновидности косвенной адресации

Простая косвенная адресация

Простая косвенная адресация применяется для копирования данных из SRAM в РОН одной из команд ld Rd,X/Y/Z, а также для пересылки в обратном направлении st X/Y/Z,Rr. В 2-байтовых регистрах-указателях X,Y,Z содержится адрес ячейки приемника либо источника в диапазоне 0…65535.

Косвенная адресация с преддекрементом

Этот вид адресации подобен простой косвенной адресации за исключением одного отличия. Перед выполнением операций пересылки значения индексных регистров X,Y,Z аппаратно уменьшаются на единицу, что и символизирует знак “-” в командах ld Rd,-X/-Y/-Z и st -X/-Y/-Z,Rr.

Косвенная адресация с постинкрементом

При косвенная адресации с постинкрементом значения указателей X,Y,Z аппаратно увеличивается на единицу (знак “+” перед указателями) после пересылки байта командами ld Rd,X+/Y+/Z+ и st X+/Y+/Z+,Rr.

Относительная косвенная адресация

Относительная косвенная адресация также используется для пересылки данных между РОН и SRAM. Однако адрес ячейки памяти определяется здесь как сумма содержимого указателей Y,Z и фиксированного смещения q. Для пересылки байта из SRAM в РОН применяются команды ldd Rd,Y+q/Z+q, а для пересылки в обратном направлении st Y+q/Z+q,Rr. Величина q может лежать в пределах 0…63.

Лекция 3.

Тема: Программирование UART/USART

Аппаратная часть

Микроконтроллер ATmega8535 имеет в своем составе модуль универсального синхронно/асинхронного приемопередатчика - USART. С его помощью между компьютером и микроконтроллером можно организовать обмен данными по последовательному каналу. Раньше в компьютерах для этих целей использовались COM порты, однако на современных машинах они уже большая редкость. Если на вашем компьютере все-таки есть такой порт, то для подключения микроконтроллера, понадобится преобразователь уровней ТТЛ – RS232. Его можно собрать на микросхеме MAX232AEPE.

Если COM порта нет, будем подключаться к USB. Для облегчения работы с этим интерфейсом и для поддержки старых устройств, использовавших RS232, производители микросхем выпускают специальные USB-UART преобразователи. Один из вариантов подобного преобразователя представлен на схеме ниже (микросхема FT232BM). При подключении его к компьютеру, система попросит драйвера. Их можно скачать на сайте производителя.

Схема для нашего примера (без переходника)

Программная часть

Для работы с USART`ом нам нам понадобятся 4 функции. Три пользовательские, которые мы сможем вызывать:

Функция инициализации

Функция отправки символа

Функция чтения приемного буфера

И одна для обработки прерывания USART`а:

Обработчик прерывания по завершению приема

Функция инициализации

Как и любой другой периферийный модуль, USART перед использованием нужно настроить. Для этого в микроконтроллере ATmega8535 есть 5 регистров – UBRRH, UBRRL, UCSRA, UCSRB, UCSRC.

```
//инициализация usart`a
void USART_Init(void)
{
 UBRRH = 0;
```

```
  UBRRL = 51; //скорость обмена 9600 бод
  //разр. прерыв при приеме, разр приема, разр передачи.
  UCSRB = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
  //обращаемся к регистру UCSRS, размер слова – 8 бит
  UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
}
```

Принятый символ USART модуль сохраняет в регистре данных UDR. Оттуда мы его потом переписываем в буфер. Выполняется это в прерывании, а для этого оно должно быть разрешено. Устанавливаем бит – RXCIE – разрешение прерывания по завершению приема в единицу.

Биты TXCIE и UDRIE – разрешают прерывания по завершению передачи и прерывания при очистке регистра данных соответственно. Эти прерывания мы сейчас использовать не будем.

Модуль USART состоит из трех основных частей – блока тактирования, блока передатчика и блока приемника. Для разрешения работы приемника и передатчика нужно установить в единицу биты RXEN и TXEN соответственно.

Размер слова данных передаваемых/принимаемых модулем USART может варьироваться от 5 до 9 разрядов и определяется битами UCSZ2..UCSZ0. В регистре UCSRB находится только бит UCSZ2. Остальные биты находятся в регистре UCSRC. Мы будем использовать 8-ми разрядные слова, а значит эти три бита должны быть установлены так - 011.

Биты RXB8 и TXB8 – это 8-ой разряд принимаемых/передаваемых данных соответсвенно. Они используются, если размер слова данных – 9 бит.

```
//разр. прерывания при приеме, разр. приема, разр. передачи
UCSRB = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);
```

У микроконтроллера ATmega8535 регистр UCSRC размещен по тому же адресу что и регистр UBRRH (это тоже регистр модуля USART, вместе с регистром UBRRL он определяет скорость обмена), поэтому при обращении к ним нужно выполнять ряд дополнительных действий для выбора конкретного регистра. За это отвечает бит URSEL. Если он установлен в 1 обращение производится к UCSRC. Устанавливаем его.

UMSEL – определяет режим работы модуля USART – синхронный, асинхронный. У нас режим работы асинхронный, разряд должен быть сброшен.

UPM1, UPM0 – определяют режимы работы схем контроля и формирования четности. Используются для повышения надежности передачи данных. В нашем случае контроль четности выключен – биты должны быть сброшены.

USBS – количество стоп битов. Для нашего случай бит сброшен.

UCSZ1, UCSZ0 – формат посылок. О них мы уже говорили выше. Для 8-ми разрядного слова эти биты должны быть установлены в единицы.

UCPOL – полярность тактового сигнала. Пропускаем мимо, потому что разряд используется только при работе в синхронном режиме.

```
//обращаемся к регистру UCSRS, размер слова – 8 бит
UCSRC = (1<<URSEL)|(1<<UCSZ1)|(1<<UCSZ0);
UBRR – UBRRH:UBRRL
```

Скорость обмена данными определяется содержимым регистра UBRR. Это 12 разрядный регистр и он физически размещается в двух регистрах ввода/вывода - UBRRH:UBRRL. Скорость обмена выбирается из ряда стандартных значений, в нашем примере она равна 9600 бод. Значение UBRR для обычного асинхронного режима (есть еще асинхронный режим с удвоенной скоростью обмена) вычисляется по формуле:

UBRR = (Fck/(16*BAUD)) – 1

где Fck – тактовая частота микроконтроллера, BAUD скорость обмена в бодах

Для нашего случая UBRR = 8000000/(16*9600) – 1 = 51. Это округленное значение, поэтому реальная скорость обмена будет отличаться от 9600. Рекомендуется использовать такие значение регистра UBRR, при которых получаемая скорость передачи отличается от требуемой на величину < 0.5%. Большая ошибка будет снижать помехозащищенность линии передачи.

```
UBRRH = 0;
UBRRL = 51; //скорость обмена 9600 бод
```

UCSRA

Этот регистр в основном содержит флаги, устанавливаемые самим модулем USART. Единственный флаг, который нам понадобится в программе – UDRE – флаг опустошения регистра данных. Он

устанавливается в 1 при пустом буфере передатчика. Установленный флаг означает, что в регистр данных можно загружать новое значение.

Функция посылающая символ

```
//отправка символа по usart`y
void USART_SendChar(unsigned char sym)
{
  while(!(UCSRA & (1<<UDRE)));
  UDR = sym;
}
```

Передача данных USART`ом инициируется записью данных в буферный регистр передатчика – регистр данных UDR. (Работа передатчика естественно должна быть разрешена). Но перед тем как записать данные, нужно убедиться что передатчик освободился и готов к получению нового слова данных. Для этого в цикле while мы проверяем бит UDRE регистра UCSRA.

Обработчик прерывания по завершению приема

```
 //прием символа по usart`y в буфер
#pragma vector=USART_RXC_vect
__interrupt void usart_rxc_my(void)
{
  usartRxBuf = UDR;
}
```

Когда модуль USART принял данные, вызывается соответствующее прерывание. В прерывании мы переписываем содержимое регистра UDR в буфер usartRxBuf. Старое значение буфера при этом затрется, но так как мы используем медленный посимвольный обмен, потерь данных не будет

Функция чтения приемного буфера

```
//чтение буфера
unsigned char USART_GetChar(void)
{
  unsigned char tmp = usartRxBuf;
  usartRxBuf = 0;
  return tmp;
```

```
}
```

Копируем значение буфера в локальную переменную, очищаем его. Локальную переменную возвращаем.

Необходимый минимум готов. Оформляем его в виде программного модуля.

usart.h

usart.c

Основная программа

Основная программа будет выглядеть так. Надеюсь дополнительных комментариев она не требует.

```
//*****************************************
//  Author(s)...: Pashgan    http://ChipEnable.Ru
//  Target(s)...: ATMega8535
//  Compiler....: IAR EWA 5.11A
//  Description.: UART/USART. Получение первых результатов
//*****************************************
#include <ioavr.h>
#include "lcd_lib.h"
#include "usart.h"

int main( void )
{
  unsigned char sym;

  USART_Init();
  LCD_Init();
  __enable_interrupt();
  LCD_SendString("uart:");

  while(1){
    sym = USART_GetChar(); //читаем буфер
```

```
    if (sym){                    //если что-то приняли, то
      LCD_Goto(6,0);
      LCD_WriteData(sym); //отображаем на lcd принятый символ
      USART_SendChar('O'); //отвечаем компу "Ok "
      USART_SendChar('k');
      USART_SendChar(' ');
    }
  }
  return 0;
}
```

Для проверки результатов работы микроконтроллера потребуется программа - терминал. Вы можете скачать ее в разделе полезный софт. Если лень собирать схему и возиться с реальным железом, можно проверить программу в Proteus`e.

Практическая работа №1

Тема: Общее устройство МК, организация памяти

Цель работы: изучить структуру, организацию памяти ARM микроконтроллеров семейства Cortex-M3, назначение выводов микросхемы и технические характеристики на примере контроллера семейства STM32F10xx.

Ход работы

Теоретические сведения

Что такое микропроцессор и как он работает

Процессор – это электронное программно управляемое устройство, предназначенное для обработки цифровой информации и выполненное в виде одной или нескольких БИС.

Из определения следует несколько важных свойств процессора:

1) процессор – это цифровая микросхема, полупроводниковое устройство. Его работа подчиняется тем же законам, что и других цифровых микросхем;

2) процессор – это устройство, которое работает по программе, т. е. алгоритм его действия заложен не в структуре микросхемы, а в последовательности команд, находящихся в памяти.

Вся информация, обрабатываемая процессором, делится на два вида: данные и команды.

Данные – это числовая информация, значения которой несут какие-либо сведения.

Команда – двоичное слово, которое, будучи помещенным в специальный регистр – регистр команд, инициирует выполнение действий по элементарной обработке информации.

Программа – это совокупность команд, выполняемых процессором.

Большинство современных процессоров, в той или иной мере, функционируют в соответствии с принципами фон Неймана. Цикл фон Неймана состоит из нескольких стадий:

– выборка команды – из памяти считывается численный код выполняемой команды;

– увеличение счетчика команд на 1; счетчик команд – специальный регистр процессора, хранящий адрес обрабатываемой команды;

– дешифрация команды – поскольку команды, как правило, хранятся в закодированном виде, то их необходимо расшифровать и определить, какие

действия должен выполнять процессор; эту операцию осуществляет дешифратор команд;

– выполнение команды – после того как стало известно, что процессор должен делать и какие данные он должен обрабатывать, происходит исполнение элементарной операции по обработке информации.

Обзор архитектуры ARM микроконтроллеров семейства Cortex

Общие сведения.

Рынок микроконтроллеров поистине огромен – ежедневно в мире продаются десятки миллионов данных устройств. На этом рынке идёт непрерывная конкурентная борьба между различными производителями, моделями и архитектурами микроконтроллеров. Рост запросов со стороны промышленного сектора вызвал потребность в более производительных микроконтроллерах; в частности, возникла необходимость в микроконтроллерах, которые при той же частоте или потребляемой мощности выполняли бы большее число операций. Кроме того, микроконтроллеры становятся всё более «коммуникабельными», используя для связи с окружающим миром шину USB, Ethernet или радиоканал, и вполне естественно, что для поддержки этих каналов связи и развитых периферийных устройств требуются дополнительные вычислительные ресурсы. Одновременно растёт сложность самих приложений, что обусловлено использованием более изощрённых пользовательских интерфейсов, необходимостью поддержки мультимедиа и увеличением функциональности устройств. Все большую популярность среди разработчиков микропроцессорных устройств завоевывают микроконтроллеры, построенные на базе ARM-архитектуры. Так, например, подавляющее большинство мобильных устройств (мобильные телефоны, планшеты) выпускаются с контроллерами, построенными на базе данной архитектуры. И все большую популярность завоевывают такие устройства на рынке промышленных систем автоматизации. Главные «прорывные» разработки ARM в сфере промышленного применения – это 32-разрядное ядро Cortex-M0, дающее скачок производительности простых приложений при сохранении цены 8-разрядных решений, и 32-разрядное ядро Cortex-M4, чьи богатые вычислительные возможности дают революционный выигрыш в приложениях, где нужна быстрая обработка входного аналогового сигнала с широким динамическим диапазоном (промышленное и лабораторное оборудование, медицинская электроника, обработка изображения или звука), шифрование информации и т. д.

При этом стоит отметить, что в отличие от многих других полупроводниковых компаний компания ARM не занимается ни изготовлением процессоров, ни продажей микросхем. Вместо этого ARM предлагает своим бизнес- партнерам, в числе которых большинство ведущих

мировых полупроводниковых компаний, лицензии на использование разработанных ею процессорных ядер. На основе дешевых и экономичных решений от ARM её партнёры создают собственные процессоры, микроконтроллеры и системы на кристалле. На сегодняшний день партнёры компании ARM ежегодно поставляют более двух миллиардов процессоров ARM. И в этой связи не стоит путать, например, такие понятия, как «Процессор Cortex-M3» и «Микроконтроллер с ядром Cortex-M3». Процессор Cortex-M3 является центральным процессором микроконтроллера, т. е. всего лишь одним, хотя и центральным, и наиболее важным, из его многочисленных узлов.

Обзор семейства Cortex.

Семейство ARM Cortex – новое поколение процессоров, которые выполнены по стандартной архитектуре и отвечают различным технологическим требованиям. В отличие от других ЦПУ ARM семейство Cortex является завершенным процессорным ядром, которое объединяет стандартное ЦПУ и системную архитектуру. В основе процессора Cortex лежит архитектура ARMv7, доступная в трех главных профилях:

1) профиль A (ARMv7-A) – для высокопроизводительных решений;

2) профиль R (ARMv7-R) – для реально-временных применений;

3)профиль M (ARMv7-M) – для чувствительных к стоимости и микроконтроллерных систем.

Как уже было указано, основой любого микроконтроллера с ядром Cortex-M3 является процессор Cortex-M3. Он представляет собой стандартизованный микроконтроллер, интегрирующий 32-битное ЦПУ (ядро процессора CM3Core), шинную структуру (Bus Matrix), контроллер вложенных прерываний (NVIC), отладочную систему (DWT и ITM) и предопределенную организацию памяти. Четыре компонента, ETM (Embedded Trace Macrocell), TPIU (Trace Port Interface Unit), SW / SWJ-DP (Serial Wire / Serial Wire JTAG Debug Port) и ROM Table (таблица постоянной памяти), находятся вне уровня Cortex-M3, потому что они или опциональны, или есть гибкость в их реализации и использовании.

Организация памяти.

Данные, обрабатываемые процессором, необходимо где-то хранить. Для этих целей микропроцессоры Cortex-M3 могут использовать регистры, являющиеся частью ядра, или запоминающие устройства (ЗУ), входящие в состав микроконтроллера.

Карта памяти. Процессор Cortex-M3 является стандартизованным

микроконтроллерным ядром и поэтому его карта памяти четко расписана. Несмотря на использование нескольких внутренних шин, адресное пространство линейное и имеет размер 4 Гбайт.

Первые 1 Гбайт памяти разделены равномерно между областью кода программы (Code) и областью статического ОЗУ (SRAM). Пространство кода программы оптимизировано для работы с шиной ICode. Аналогично пространство статического ОЗУ доступно через шину DCode. Несмотря на то, что в области статического ОЗУ поддерживается загрузка и исполнение инструкций, их выборка осуществляется через системную шину, что требует дополнительного состояния ожидания.

Таким образом, выполнение кода программы из статического ОЗУ будет более медленным, чем из встроенной Flash-памяти, расположенной в области кода программы.

Следующие 0,5 Гбайт памяти – область встроенных УВВ (Peripheral), где находятся все предоставляемые пользователю производителем микроконтроллера устройства ввода/вывода. Первые 1 Мбайт в областях статического ОЗУ и УВВ являются бит-адресуемыми. Для этого используется метод bit-banding. Таким образом, все данные, хранящиеся в этих областях, могут обрабатываться как пословно, так и побитно.

Следующие 2 Гбайт адресного пространства выделены для внешних статического ОЗУ (External RAM) и устройств ввода/вывода (External device).

Последние 0,5 Гбайт зарезервированы для системных ресурсов процессора Cortex (Private peripheral bus – Internal – локальная шина УВВ внутренняя, Private peripheral bus – External – локальная шина УВВ внешняя) и будущих расширений процессора Cortex (Vendor specific – специфическая область производителя. В данной области памяти производитель микроконтроллера может размещать свою информацию). Все регистры процессора Cortex расположены по фиксированным адресам во всех Cortex-микроконтроллерах. Благодаря этому облегчается портирование программ между различными Cortex-микроконтроллерами разных производителей.

Контрольные вопросы

1 Перечислите характерные черты ARM микроконтроллеров семейства Cortex-M3.

2 Какие структурные элементы входят в состав процессора Cortex-M3?

3 Назовите назначение структурных элементов, входящих в состав процессора Cortex-M3.

4 Каким образом организована память процессора Cortex-M3?

5 Укажите назначение регистров процессора Cortex-M3.

6 Поясните формат слова состояний xPSR.

7 Поясните особенности использования стека в процессоре Cortex-M3.

8 Перечислите основные особенности микроконтроллера STM32F103R8.

9 Назовите объем резидентной памяти данных и программ микроконтроллера STM32F103R8.

10Укажите отличительные особенности организации памяти микроконтроллера STM32F103R8.

11 Перечислите назначение выводов микроконтроллера.

12 Перечислите альтернативные функции параллельных портов.

Практическая работа №2

Тема: Система команд AVR

Цели работы:

- Знакомство с основными возможностями среды проектирования AVR-Studio (создание проекта, включение в него файлов; компиляция; запуск программы на исполнение в симуляторе; использование основных отладочных ресурсов – пошагового исполнения, установки точек останова, просмотра и изменения состояния регистров, памяти).
- Знакомство с арифметическими командами AVR (состав, особенности использования, влияние на флаги регистра состояния процессора, способы повышения разрядности арифметических операций).

Ход работы

1. Ознакомьтесь с теоретической частью лабораторной работы.
2. Разработайте блок-схемы программ для заданий 2-6.
3. Подготовьтесь к ответу на контрольные вопросы 6-18.
4. Получите допуск к выполнению практической части лабораторной работы.
5. Выполните задания, подготовьтесь к ответу на контрольные вопросы 1-5.
6. Защитите лабораторную работу.

Краткие теоретические сведения

Информация об основных функциях AVR Studio находится в файле справки среды AVR Studio (раздел AVR Studio). Информация о структуре программ на языке Ассемблера AVR, выражениях и директивах ассемблера также находится в файле справки среды AVR Studio (раздел AVR Assembler).

Систему команд At90S2313 можно разделить на следующие основные группы:

команды пересылок данных;

арифметические и логические команды и команды манипуляций с битами;

команды передачи управления;

специальные команды (NOP, WDR, SLEEP).

Данная лабораторная работа посвящена знакомству с арифметическими командами.

Команды сложения At90S2313: сложение двух регистров (ADD), сложение двух регистров с учетом флага переноса (ADC), сложение слова с константой (ADIW). Эти команды устанавливают флаг переноса C, если при их выполнении произошел перенос из старшего, 7-го (для команды ADIW – 15-го) разряда результата, и сбрасывают, если перенос не произошел. Флаг C позволяет организовать сложение чисел любой разрядности, пользуясь командами ADD и ADC. Следующий пример показывает сложение двух 24-разрядных чисел, хранящихся в регистрах r2:r1:r0 и r5:r4:r3; результат сохраняется в регистрах r2:r1:r0.

```
add r0,r3  ;сложение младших байтов операндов

adc r1,r4  ;сложение средних байтов операндов, учитывая перенос из 7-го бита

adc r2,r5  ;сложение старшие байтов операндов, учитывая перенос из 14-го бита
```

Приведенный фрагмент корректен для чисел как со знаком, так и без знака. Для представления знаковых чисел, как и в большинстве целочисленных процессоров, в AVR используется дополнительный код. Вообще, для выполнения знаковых и беззнаковых операций сложения и вычитания используются одни и те же команды. Отличие имеется лишь в интерпретации результата. Так, флаг переполнения V устанавливается при выполнении операций сложения в том случае, если при одинаковых знаковых разрядах операндов результат имеет другой знак. Иначе, флаг V сбрасывается. Если рассматривать операнды и результат как знаковые, флаг V будет установлен, если результат больше максимального или меньше минимального для данной разрядной сетки числа со знаком. Например, выполняем команду ADD – сложение 8-разрядных регистров. Для 8-разрядной сетки минимальное число равно –128, максимальное равно +127. При попытке сложить –15 и –120, или, скажем, +54 и +93, очевидно, будет происходить переполнение и устанавливаться флаг V.

Флаг отрицательного результата N будет устанавливаться, если старший, знаковый, разряд результата равен единице (т.е., результат – отрицательный, если рассматривать его как знаковое число).

Флаг знака S принимает значение исключающего ИЛИ флагов N и V: NÅV. То есть, при отсутствии переполнения (V=0) флаг S дублирует флаг отрицательного результата N; при наличии переполнения – содержит инверсию флага N, или, что то же самое – знак 9-разрядного знакового результата операции.

Флаг полупереноса H по своей сути аналогичен флагу C – он устанавливается при наличии переноса из 3-го разряда результата. Этот флаг может быть использован для реализации операций с 4-разрядными числами, а также с числами в BCD-формате. В следующем фрагменте показано

одновременное сложение двух пар 4-разрядных чисел. При наличии переноса из результата сложения младших тетрад в результат сложения старших производится коррекция.

```
        add r16,r17
        brhc m1
        subi r16,0x10
m1:
  …
```

Флаг Z устанавливается в случае, когда восемь (для ADIW – шестнадцать) младших разрядов результата равны нулю.

Особенностью команды ADIW являются ограничения:

на значение константы (она должна быть в диапазоне от 0 до 63);

на используемые регистры (применяются только пары регистров r25:r24, r27:r26, r29:r28 и r31:r30; в команде указывается младший регистр пары).

Корректное использование команды ADIW:

adiw r24,15 ;к содержимому 16-разрядного регистра r25:r24 прибавляется 15

adiw r30,49 ;к содержимому 16-разрядного регистра r31:r30 прибавляется 49

Некорректное использование команды ADIW:

adiw r24,75 ;значение константы превышает 63

adiw r27,15 ;недопустимый регистр

Команды вычитания At90S2313: вычитание двух регистров без учета заема (SUB), вычитание двух регистров с учетом заема (SBC), вычитание константы из регистра без учета заема (SUBI), вычитание константы из регистра с учетом заема (SBCI), вычитание константы из слова (SBIW).

Эти команды используют флаг C как флаг заема: он устанавливается, если уменьшаемое меньше вычитаемого (т.е., произошел заем из 8-го, следующего за старшим, разряда вычитаемого). Аналогично случаю с операциями сложения, при вычитании флаг C также позволяет наращивать разрядность. Пример показывает вычитание двух 24-разрядных чисел (r5:r4:r3 из r2:r1:r0); результат сохраняется в регистрах r2:r1:r0.

```
sub r0,r3  ;вычитаем младшие байты операндов
```

sbc r1,r4 ;вычитаем средние байты операндов, учитывая заем из 8-го бита

sbc r2,r5 ;вычитаем старшие байты операндов, учитывая заем из 15-го бита

Команды SUBI и SBCI позволяют вычитать константы из регистров. Ограничений на значения констант (в отличии, например, от команд ADIW и SBIW) в этих командах нет (т.е., константы могут принимать значения от 0 до 0xFF). Однако, эти команды применимы только к старшей половине регистрового файла. С помощью этих команд можно реализовать и операцию сложения регистров с константами, поскольку R+K°R–(–K). Например, 16-разрядное сложение регистровой пары r17:r16 и числа 51 будет выглядеть следующим образом:

subi r16,LOW(-51) ;

sbci r17,HIGH(-51) ;

Флаг переполнения V устанавливается при наличии заема из 7-го, знакового, бита результата. Флаги N, Z и S при вычитании ведут себя также как и при операциях сложения. Флаг H устанавливается при наличии заема из 4 бита уменьшаемого.

Команда 16-разрядного вычитания константы из регистра SBIW имеет те же ограничения на регистры и константы, что и команда сложения ADIW.

Еще две арифметические команды – INC и DEC отвечают, соответственно, за увеличение и уменьшение значений регистров на единицу:

inc r16 ;r16 <- r16+1

dec r0 ;r0 <- r0–1

В отличие от команд сложения и вычитания, команды инкремента и декремента не изменяют состояния флагов C и H, поэтому, например, попытка декрементировать 16-разрядную переменную r17:r16 следующим образом будет некорректна:

dec r16

sbci r17,0

Корректным будет, например, такой вариант:

subi r16,1

sbci r17,0

Остальные флаги при выполнении команд инкремента и декремента ведут себя так же, как и при обычных операциях сложения и вычитания.

Несколько команд позволяют анализировать значения одного или двух регистров, не изменяя эти значения: сравнение двух регистров (CP), сравнение двух регистров с учетом флага заема (CPC), сравнение регистра с константой (CPI), проверка содержимого регистра на равенство нулю (TST).

Команды сравнения изменяют флаги так же, как и операции вычитания, но не изменяют значения регистров. Команда CPI работает только со старшей половиной регистрового файла.

Время выполнения всех арифметических команд составляет один такт синхронизации процессора. Исключением являются команды, работающие со словами (ADIW и SBIW) – они выполняются за два такта.

Задания:

Создайте в вашем рабочем каталоге новый проект для микроконтроллера At90S2313. Создайте новый ассемблерный файл и включите его в состав проекта. Напишите простейшую программу, подобную показанной в примере 1. Откомпилируйте её и загрузите на выполнение в симуляторе. Пользуясь справочной системой AVR-Studio, на примере данной программы научитесь пользоваться основными функциями отладчика:

загрузка, запуск и останов программы, сброс микроконтроллера;

пошаговое выполнение программы;

задание точек останова;

просмотр состояния процессора;

просмотр и изменение состояния регистров, памяти данных и Flash-памяти программ.

Напишите программу а) сложения, б) вычитания двух 32-разрядных регистровых переменных.

Напишите программу сложения 16-разрядной и 8-разрядной регистровых переменных для случаев, когда обе переменные а) знаковые и б) беззнаковые.

Напишите программу сложения 32-разрядной регистровой переменной с константой.

Напишите программу, которая находит модуль разности двух 8-разрядных чисел а) без знака, б) со знаком.

Напишите программу, которая вычисляет сумму r1:r0 и r3:r2, если r1:r0==r3:r2, разность r1:r0 и r3:r2, если r1:r0 и r3:r2 имеют разные знаки, иначе вычисляет выражение r1:r0+(0xFFFF-r3:r2).

Пример 1:

Показывает структуру простейшей программы на языке Ассемблера для AVR-микроконтроллеров.

```
.INCLUDE "..appnotes2313def.inc" ;подключение файла со
;спецификацией регистров ввода/вывода
.CSEG                ;сегмент кода
rjmp Reset           ;вектор прерывания, вызываемого по сбросу микроконтроллера
Reset:               ;начало программы
    ser r17          ;установка всех битов r17 в единицу
    mov r0,r17       ;копирование содержимого r17 в r0
    ldi r16,0x55  ;загрузка в r16 числа 0x55
    subi r16,0x11    ;r16<-r16 - 0x11
    add r16,r0       ;r16<-r16 + r0
rjmp PC;бесконечный цикл
```

Контрольные вопросы и задания:

1. Какими средствами симулятора AVR-Studio можно воспользоваться для просмотра значений регистров-указателей X, Y, Z?
2. Необходимо оценить время исполнения некоторого фрагмента программы. Какими средствами симулятора AVR-Studio можно воспользоваться для решения этой задачи?
3. При симуляции программы из примера 2 необходимо заполнить массив значениями, отличными от нуля. Какими средствами симулятора Вы для этого воспользуетесь?
4. Покажите средствами AVR-Studio, что команда очистки регистра clr Rd на самом деле имеет тот же машинный код, что и команда исключающего ИЛИ eor Rd, Rd.
5. Оценка времени исполнения некоторого фрагмента программы показала значение 10 мкс при тактовой частоте микроконтроллера 2 МГц. Какое время займёт её исполнение при тактовой частоте 10 МГц?
6. Для чего в конце программы нужен оператор rjmp PC ?
7. Укажите общие черты и отличия процессора, микропроцессора и микроконтроллера.
8. Особенности основных современных типов ПЗУ (Flash, EPROM, EEPROM, EPROM с УФ стиранием).
9. Гарвардская и фон-неймановская архитектуры компьютеров.

10. Представление целых чисел в ЭВМ: прямой и дополнительный коды. Правила операций сложения и вычитания чисел в прямом и дополнительном кодах.
11. Арифметические флаги процессорного ядра AVR, их назначение.
12. Разница между флагами переноса и переполнения.
13. Наращивание разрядности операций сложения.
14. Наращивание разрядности операций вычитания.
15. Особенности реализации в AVR команд инкремента и декремента.
16. Реализация операции сложения с константой через команду вычитания.
17. Как с помощью 8-разрядных команд сравнения организовать сравнение переменных большей разрядности? Переменной и константы?
18. Как корректно сложить (вычесть) два операнда разной разрядности, если они знаковые (беззнаковые)?

Практическая работа №3

Тема: Программирование UART/USART

1 Цель работы

1. Изучить схему подключения микроконтроллера к компьютеру.
2. Изучить особенности работы последовательного асинхронного порта UART.
3. Освоить методику расчета скорости последовательного порта.
4. Изучить особенности программирования UART.
5. Изучить способы отладки программ на учебном лабораторном стенде LESO1.

2 Предварительная подготовка к работе

1. По конспекту лекций и рекомендуемой литературе изучить принцип работы последовательного асинхронного порта UART.
2. По документации изучить особенности порта UART микроконтроллера ADuC842.
3. Составить алгоритм работы программы, соответственно заданию.
4. Составить программу на языке программирования С.

3 Краткие теоретические сведения

3.1 Последовательный асинхронный интерфейс UART

Последовательный интерфейс использует одну сигнальную линию для передачи данных, по которой биты информации передаются друг за другом последовательно. При последовательной передаче сокращается количество сигнальных линий, что упрощает разводку проводников на печатной плате, уменьшает габариты устройства и позволяет делать более помехозащищенные интерфейсы. При последовательной передаче каждый информационный бит должен сопровождаться импульсом синхронизации — стробом. Если импульсы синхронизации передаются от одного устройства к другому по выделенной линии, то такой интерфейс называют синхронным, в этом случае генератор синхронизации располагается на стороне устройства инициирующего передачу. Если же приемник и передатчик содержат каждый свой генератор синхроимпульсов, работающий на одной частоте, то такой интерфейс называется асинхронным. Получается, что приемник информации сам вырабатывает синхроимпульсы.

Типичный представитель асинхронного последовательного интерфейса — UART (Universal Asynchronous Receiver-Transmitter — универсальный асинхронный приёмопередатчик).

При передаче по интерфейсу UART каждому байту данных предшествует СТАРТ-бит, сигнализирующий приемнику о начале посылки, за СТАРТ-битом следуют биты данных. Завершает посылку СТОП-бит, гарантирующий паузу между посылками. СТАРТ-бит следующего байта посылается в любой момент после СТОП-бита, то есть между передачами возможны паузы произвольной длительности. СТАРТ-бит, обеспечивает простой механизм синхронизации приемника по сигналу от передатчика. Внутренний генератор синхроимпульсов приемника использует счетчик-делитель опорной частоты, обнуляемый в момент приема начала СТАРТ-бита. Этот счетчик генерирует внутренние стробы, по которым приемник фиксирует последующие принимаемые биты.

3.2 Особенности работы UART микроконтроллера ADuC842

В микроконтроллере ADuC842 последовательный порт UART является полнодуплексным — позволяет передавать и принимать данные одновременно. В рассматриваемом микроконтроллере прием через последовательный порт буферизирован: порт может начать принимать байт данных еще до того как предыдущий байт будет считан из буферного регистра приемника. Однако, если до конца приема предыдущий байт не будет считан из буфера, он будет потерян: новый принятый байт перезапишет старый.

Принимаются и передаются данные по разным линиям, передача происходит через вывод TxD микроконтроллера (Transmitter Data — передатчик данных), прием — через RxD (Receiver Data — приемник данных). Физически выводы RxD и TxD совмещены с выводами третьего параллельного порта P3.0 и P3.1 соответственно.

Программное взаимодействие с последовательным портом UART осуществляется через регистры специальных функций (SFR) SBUF и SCON. Через SBUF (Serial buffer — последовательный буфер) осуществляется доступ к регистрам приемника и передатчика последовательного порта. Когда программно производится запись в SBUF, то данные загружаются в регистр передатчика, когда же программой происходит чтение SBUF, то осуществляется доступ к регистру приемника. Физически регистры приемника и передатчика разделены. SFR адрес — 0x99.

SCON — регистр конфигурации и управления последовательным портом микроконтроллера.

SFR адрес — 0x98. Значение после подачи питания 0x00. Регистр имеет побитовую адресацию.

Таблица 1 – Назначение бит регистра SCON

<table>
<tr><th>номер</th><th>мнемоника</th><th>описание</th></tr>
<tr><td>7</td><td>SM0</td><td rowspan="2">SM1, SM0 биты определяют режим работы последовательного порта.
<table>
<tr><th>SM0</th><th>SM1</th><th>Выбранный режим</th></tr>
<tr><td>0</td><td>0</td><td>режим 0: синхронный режим с фиксированной скоростью $f_{core/2}$.</td></tr>
<tr><td>0</td><td>1</td><td>режим 1: 8-битный асинхронный режим с настраиваемой скоростью передачи данных.</td></tr>
<tr><td>1</td><td>0</td><td>режим 2: 9-битный асинхронный режим с фиксированной скоростью $f_{core}/32$ или $f_{core}/16$.</td></tr>
<tr><td>1</td><td>1</td><td>режим 3: 9-битный асинхронный режим с настраиваемой скоростью передачи данных.</td></tr>
</table></td></tr>
<tr><td>6</td><td>SM1</td></tr>
<tr><td>5</td><td>SM2</td><td>Бит управления режимом приемопередатчика. Устанавливается программно для запрета приема сообщения, в котором девятый бит имеет значение “0”. Такой режим используется при реализации сетевого протокола в многопроцессорной системе.</td></tr>
<tr><td>4</td><td>REN</td><td>Бит разрешения приема данных через последовательный порт. Устанавливается программно, для разрешения приема.</td></tr>
<tr><td>3</td><td>TB8</td><td>9-ый бит передатчика последовательного порта. Данные загруженные в TB8 передаются девятым битом. Это актуально для режима работы UART 2 и 3.</td></tr>
<tr><td>2</td><td>RB8</td><td>9-ый бит приемника последовательного порта. В режиме работы 2 и 3 в этот бит загружается принятый девятый бит данных. В режиме 1 в этом бите хранится СТОП-бит.</td></tr>
<tr><td>1</td><td>TI</td><td>Флаг прерывания передатчика последовательного порта. Устанавливается аппаратно при окончании передачи байта. Флаг должен сбрасываться (запись “0”) программно.</td></tr>
<tr><td>0</td><td>RI</td><td>Флаг прерывания приемника последовательного порта. Устанавливается аппаратно при окончании приема байта. Флаг должен сбрасываться (запись “0”) программно.</td></tr>
</table>

Режим работы 0: режим 8-битного сдвигового регистра. Для выбора этого режима работы следует в биты SM0 и SM1 записать логические нули (SM1 = 1 и SM0 = 0). В этом режиме данные последовательно передаются и принимаются через вывод RxD. Вывод TxD используется для подачи тактовых импульсов. Передача инициируется любой инструкцией, записывающей данные в регистр SBUF. Передача байта данных начинается с

младшего значащего бита. Прием начинается, когда бит разрешения приема установлен в единицу (REN = 1), а флаг прерывания приемника сброшен в ноль (RI = 0). Таким образом, в данном случае последовательный порт работает в синхронном режиме, поэтому у некоторых семейств микроконтроллеров такой последовательный порт называется не UART, а USART (Universal Synchronous-Asynchronous Receiver-Transmitter — универсальный синхронный - асинхронный приемопередатчик).

Режим работы 1: асинхронный 8-битный режим с настраиваемой скоростью. Для выбора первого режима следует в бит SM0 записать единицу, а в бит SM1 — ноль (SM1 = 0 и SM0 = 1). Вывод микросхемы TxD используется для передачи информации, вывод RxD — для приема.

Передача байта информации начинается с посылки СТАРТ-бита, за ним идут восемь информационных бит, заканчивается передача СТОП-битом. Таким образом, для передачи каждого байта информации используется 10 бит. Формирование СТАРТ-бита и СТОП-бита происходит автоматически.

Скорость передачи может устанавливаться Таймером 1 или Таймером 2, или комбинацией их обоих: один — для передачи, другой — для приема. Использование таймеров 1 и 2 характерно для всего семейства MCS-51/52, правда Таймер 2 есть только в старших моделях, но в микроконтроллере ADuC842 возможна синхронизация приемопередатчика UART от специального Таймера 3, что позволяет освободить таймеры общего назначения для выполнения других функций.

Передача данных начинается, когда в регистр SBUF программно записывается передаваемое число. Данные передаются до тех пор, пока на TxD не поступит СТОП-бит, после чего флаг прерывания передатчика (TI) установится в единицу, как это показано на рисунке ниже:

Рисунок 1 – Диаграмма передачи байта

Для разрешения приема данных через последовательный порт в бит REN следует записать логическую единицу. Прием байта данных начинается с приходом на линию RxD СТАРТ-бита: перехода линии из высокого

логического уровня в низкий. По окончанию приема всего байта флаг прерывания приемника RI устанавливается в единицу, и принятый байт может быть программно считан из буферного регистра SBUF.

Режим работы 2: асинхронный 9-битный режим с фиксированной скоростью. Скорость передачи данных по умолчанию fcore/32, но если записать в бит SMOD регистра PCON логическую единицу, то скорость передачи будет удвоена: fcore/16. Для выбора второго режима следует в бит SM0 записать логический ноль, а в бит SM1 — единицу (SM1 = 1 и SM0 = 0).

В этом режиме каждая передаваемая или принимаемая посылка состоит из одиннадцати бит: СТАРТ-бит, восемь информационных бит, девятый программируемый бит и СТОП-бит. Девятый бит часто используют в качестве бита паритета при контроле четности, хотя он может быть использован для любых других целей.

Прием и передача во втором режиме осуществляется аналогично первому режиму. Девятый бит при передаче программно записывается в бит TB8 регистра SCON, при приеме девятый бит находится в RB8 регистра SCON.

Режим работы 3: асинхронный 9-битный режим с настраиваемой скоростью. Для выбора третьего режима следует в биты SM0 и SM1 записать логические единицы (SM1 = 1 и SM0 = 1). В этом режиме последовательный порт UART работает также как в режиме 2, только скорость задается таймерами 1, 2 или 3, так же, как и в режиме 1.

Во всех четырех режимах передача данных начинается любой инструкцией, записывающей в регистр SBUF число. В режиме 0 прием начинается при условии RI = 0 и REN = 1. Во всех остальных режимах прием начинается с приходом СТАРТ-бита при условии, что в бит REN записана логическая единица (REN = 1).

3.3 Расчет параметров синхронизации UART

По умолчанию последовательный порт микроконтроллера ADuC842 настроен на синхронизацию от Таймера 1. Скорость передачи UART определяется временем переполнения таймера:

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

где T_T – время срабатывания таймера.

Таймер должен быть сконфигурирован для работы в режиме с автоперезагрузкой (режим 2). Для установки такого режима в старшие 4 бита регистра **TMOD** следует записать бинарную комбинацию 0010b. В этом случае скорость передачи данных будет определяться по формуле:

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

где: f_{core} – частота ядра микропроцессора, TH1 – содержимое регистра данных **TH1**.

Из формулы 2 легко найти значение регистра **TH1**, обеспечивающего требуемую скорость:

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

Результат вычисления должен быть округлен до ближайшего целого.

Используя Таймер 1 для синхронизации UART не всегда возможно получить требуемую частоту с достаточной точностью. Например, пусть при тактовой частоте ядра микропроцессора 2097кГц (значение для ADuC842 по умолчанию), требуется получить скорость передачи 19.2 кбит/с. По формуле 3 найдем значение TH1:

$$TH1 = 256 - \frac{2097}{32 \cdot 19.2} = 252{,}59 \approx 252.$$

Используя полученное значение TH1, рассчитаем реальную скорость передачи UART:

$$BR = \frac{2097}{32 \cdot (256 - 252)} = 16.38 \left(\frac{\text{кбит}}{\text{с}}\right).$$

Реальная скорость на 14% меньше требуемой, а это значит, что передача данных невозможна. Проблема была решена добавлением специального таймера 3, специализированного для высокоточной синхронизации UART в широком диапазоне частот. Кроме того, использование специализированного таймера высвобождает таймеры общего назначения для решения других различных задач.

Таймер 3, по сути, представляет собой набор настраиваемых делителей тактовой частоты ядра, структурная схема таймера изображена на рисунке ниже:

Для управления Таймером 3 предназначены два регистра специальных функций — T3CON и T3FD. Регистр T3CON содержит бит T3EN, при записи в него логической единицы синхронизация UART будет происходить от Таймера 3, в противном случае — от Таймера 1. Младшие три бита регистра T3CON определяют двоичный делитель DIV. Дробный коэффициент деления настраивается регистром T3FD.

T3CON – регистр конфигурации Таймером 3.

SFR адрес — 0x9E. Значение после подачи питания 0x00. Регистр не имеет побитовой адресации.

Таблица 2 – Назначение битов регистра T3CON

<table>
<tr><th>номер</th><th>мнемоника</th><th>описание</th></tr>
<tr><td>7</td><td>T3EN</td><td>Разрешение Таймера 3.
Когда бит установлен (T3EN = 1) синхронизация приемника и передатчика последовательного порта происходит от Таймера 3.
Когда бит сброшен (T3EN = 0) — синхронизация от Таймера 1.</td></tr>
<tr><td>6</td><td></td><td rowspan="4">Не используются</td></tr>
<tr><td>5</td><td></td></tr>
<tr><td>4</td><td></td></tr>
<tr><td>3</td><td></td></tr>
<tr><td>2</td><td>DIV2</td><td rowspan="3">Биты целочисленного делителя DIV.
<table><tr><th>DIV2</th><th>DIV1</th><th>DIV0</th><th>DIV</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table></td></tr>
<tr><td>1</td><td>DIV1</td></tr>
<tr><td>0</td><td>DIV0</td></tr>
</table>

T3FD – регистр Таймера 3. **SFR** адрес — **0x9D**. Значение после подачи питания **0x00**. Регистр не имеет побитовой адресации.

Используя структурную схему Таймера 3 легко записать аналитическое выражение для расчета результирующей скорости последовательного порта:

$$\mathrm{BR} = \frac{2 \cdot f_{core}}{2^{\mathrm{DIV}-1} \cdot (\mathrm{T3FD} + 64)}, \quad (4)$$

где f_{core} — частота ядра микроконтроллера.

Значение делителя **DIV** можно определить по формуле 5, полученное значение следует округлить до целого вниз.

$$\mathrm{DIV} = \log_2 \frac{f_{core}}{16 \cdot \mathrm{BR}}. \quad (5)$$

Дробный делитель T3FD можно найти по формуле 6, полученное значение следует округлить до ближайшего целого.

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

Рассчитаем параметры конфигурации Таймера 3, для предыдущего примера: при тактовой частоте ядра микропроцессора 2097кГц, требуется получить скорость передачи 19.2 кбит/с.

$$DIV = \log_2 \frac{2097}{16 \cdot 19.2} = 2.771 \approx 2$$

$$T3FD = \frac{2 \cdot 2097}{2^{2-1} \cdot 19.2} - 64 = 45.219 \approx 45$$

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

Таким образом, ошибка установления скорости составляет всего 0.2%.

3.4 Особенности представления текстовой информации

В различных операционных системах для представления текстовой информации используют специальные наборы символов. Как правило, такой набор представляют в виде таблице, где каждому символу соответствует бинарная последовательность длиною в один или несколько байт. В литературе подобную таблицу символов часто называют «кодировкой». На сегодняшний день наиболее распространенным является код ASCII (American Standard Code for Information Interchange — американский стандартный код для обмена информацией), который используется для внутреннего представления символьной информации в операционной системе MS DOS, в Блокноте операционной системы Windows, а также для кодирования текстовых файлов в Интернет.

Поскольку первоначально ASCII предназначался для обмена информации, в нём, кроме информационных символов, используются символы-команды для управления связью. В приведенной таблице такие символы показаны многоточием.

Таблица 3 – Таблица ASCII

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

1.	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2.		!	"	#	$	%	&	'	(	)	*	+	,	—	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	...

Таблица кодов содержит 8 столбцов и 16 строк, каждая строка и столбец пронумерованы в шестнадцатеричной системе счисления. Шестнадцатеричное представление ASCII-кода складывается из номера столбца и номера строки, в которых располагается символ, при этом номер строки образуют первую цифру (старшие четыре бита), а номер столбца вторую цифру (младшие 4 бита). Так, например, ASCII-код символа “E” есть число 0x45, а символа “\“ — 0x5C.

Легко заметить, что в приведенной таблице представлено 128 символов, притом, что один символ кодируется байтом — восьмью битами. Дело в том, что верхние значения (128—255) могут занимать различные дополнительные символы, например, набор русского алфавита, это зависит от конкретного типа кодировки.

3.5 Способы программной реализации работы UART

Перед первым обращением к приемо-передатчику UART последовательный порт должен быть настроен: определен режим работы, выбран и настроен источник синхронизации. Режим работы UART устанавливается битами SM0 и SM1 регистра SCON. Так как регистр имеет как байтовую, так и битовую адресацию, выполнить настройку можно разными способами: записать в регистр SCON требуемое число или установить каждый бит отдельно. Источник синхронизации определяется битом T3EN регистра T3CON: если в этот бит записать логическую единицу, то синхронизация будет происходить от Таймера 3, если ничего не записывать (по умолчанию T3EN = 0), то синхронизация от Таймера 1.

При использовании Таймера 1 необходимо сконфигурировать его для работы в режиме 2 (свободнобегущий таймер с автоперезагрузкой), для этого в старшие четыре бита регистра SMOD следует записать двоичную комбинацию 0010b. Регистр счетчика TH1 определяет скорость передачи информации по UART, его значение следует рассчитать по формуле 3. После

записи TH1 таймер нужно запустить, делается это записью в бит TR1 регистра TMOD логической единицы.

При синхронизации от Таймера 3, по формулам 5 и 6 рассчитываются делители DIV и T3FD. Если запись делителя T3FD делается непосредственно в регистр T3FD, то делитель DIV определяется младшими тремя битами регистра T3CON, при этом в старший бит этого регистра (T3EN) должна быть записана логическая единица. Запуск таймера происходит автоматически.

Отправление данных по UART начинается любой командой, результат выполнения которой записывается в регистр SBUF:

```
SBUF =0x45; // отправить символ “E”
```

Можно каждый раз не пользоваться таблицей ASCII для определения кода символа, в языке «Си» для этого есть удобный инструмент: достаточно взять требуемый символ в апострофы, компилятором это будет интерпретировано как код символа.

```
SBUF ='E'; // отправить символ “E”
```

Если же требуется отправить не один символ, то прежде чем следующий код будет записан в SBUF, следует подождать, пока предыдущий символ будет отправлен. О конце передачи сигнализирует флаг TI регистра TCON, когда передача завершена, в бит TI аппаратно записывается логическая единица. Можно программно организовать в цикле проверку TI на равенство нулю, а следующий байт отправлять только тогда, когда TI окажется равен единице:

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

Аналогично выполняется и прием байта. Принятый байт может быть считан из буферного регистра лишь тогда, когда был принят последний бит и флаг приема RI установлен.

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

В приложениях, где время выполнение критично недопустимо тратить много времени при передаче на ожидание пока байт будет отправлен и буфер освободится, в этом случае можно не дожидаясь полной отправки байта приступить к выполнению дальнейшей программы, но перед отправкой

следующего байта нужно убедиться, что буфер освободился и передатчик готов к работе. Участок программы, отправляющий байт данных можно переделать следующим образом:

while(!TI); // подождать, пока буфер передачи не освободится (если занят)

SBUF = 0x45; // заполнить буфер и начать передачу

TI = 0; // сбросить флаг передачи в нуль

С таким вариантом реализации устраняются паузы на выполнение программы между передачами отдельных байтов.

Ниже на рисунке 3 приведены адреса регистров специальных функций, используемых в работе.

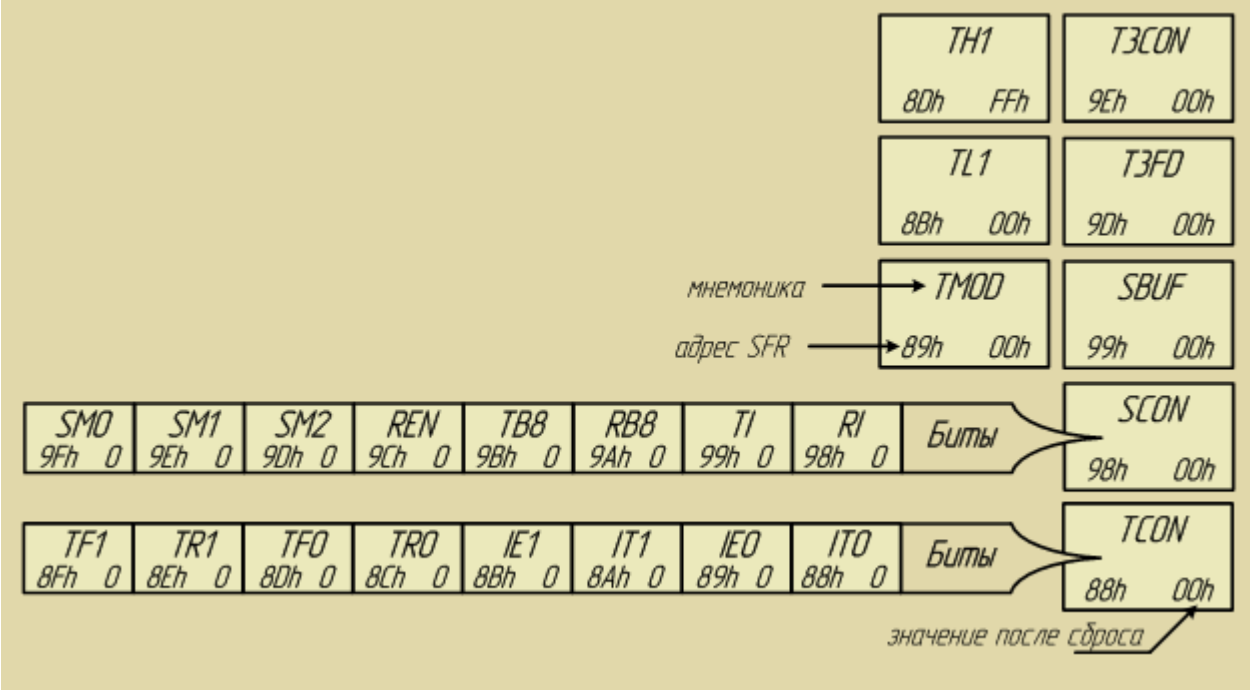

Рисунок 3 – Адреса регистров специальных функций

При написании программы следует помнить, что программа для микроконтроллера должна выполняться до отключения питания устройства и не может быть завершена. Поэтому программа должна содержать бесконечный цикл.

3.6 Взаимодействие микроконтроллера с персональным компьютером

Учебный лабораторный стенд LESO1 подключается к персональному компьютеру через микросхему преобразователя интерфейсов USB-UART. Для связи с микроконтроллером в программе загрузчика nwFlash реализован терминал. Терминал позволяет посылать через последовательный порт в микроконтроллер информацию, принимать и отображать принятую из

микроконтроллера информацию. Настройка терминала осуществляется в пункте главного меню «Опции терминала». Опции терминала позволяют:

выбрать режим отображения данных: текстовый или шестнадцатеричный, при этом изменяется также тип посылаемых данных;

выбрать кодировку ANSI (Windows-1251) или ASCII (DOS-866);

включать и выключать режим автоматической прокрутки текста;

очистить окно терминала;

сохранять принятую от микроконтроллера информацию в файл:

в том виде, как она пришла — пункт меню «Сохранить»;

в том виде, как она отображается в терминале — пункт меню «Сохранить как текст».

На рисунке 4 показана вкладка главного меню «Опции терминала».

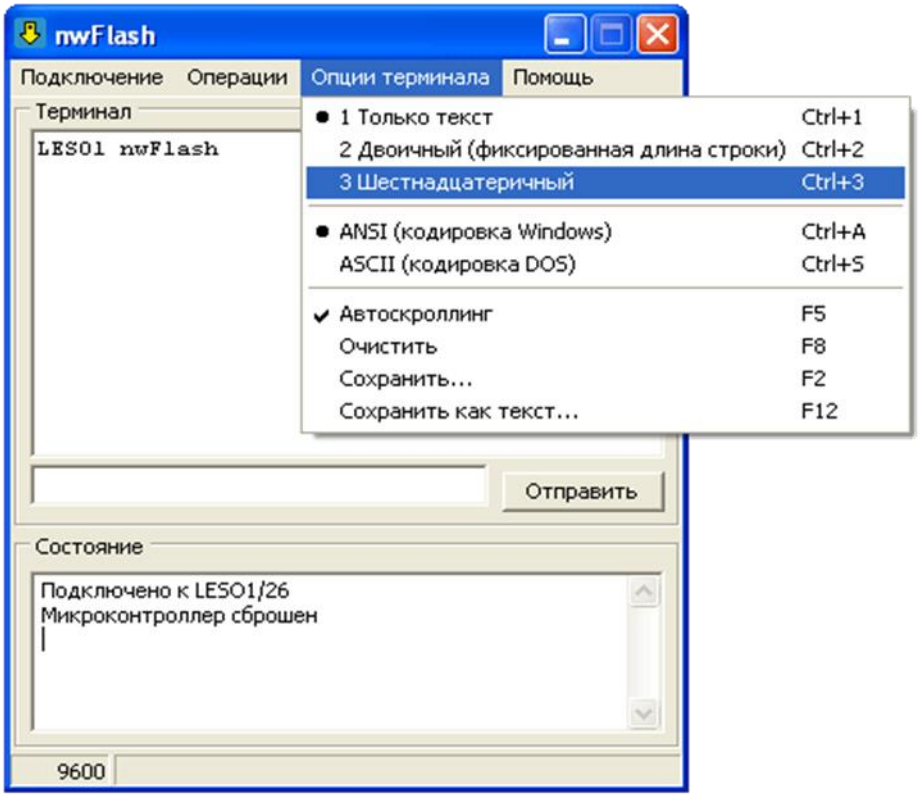

Рисунок 4 – Настройка опций терминала

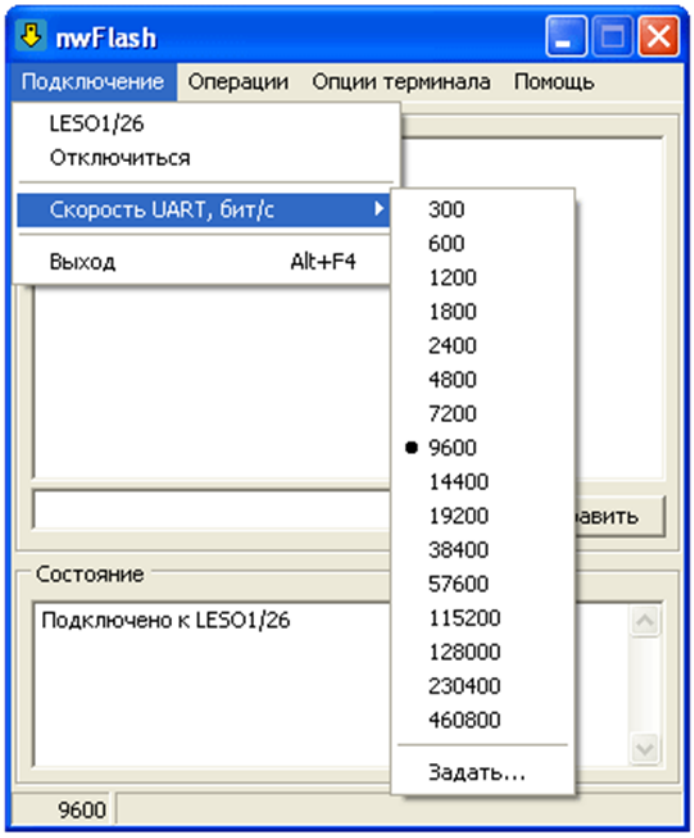

Рисунок 5 – Настройка скорости UART

При обмене данными с учебным стендом необходимо установить требуемую скорость подключения. Сделать это можно в меню «подключение», как это показано на рисунке 5.

4 Задание к работе в лаборатории

4.1 Вывод информации через послиедовательный порт

1. Разработайте алгоритм программы, передающей через последовательный порт UART данные в персональный компьютер — фамилию студента. Скорость передачи данных должна соответствовать варианту (таблица 5). Источник синхронизации UART (Таймер 1 или Таймер 3) согласовывается с преподавателем. Взаимодействие микроконтроллера с компьютером осуществляется через терминал программы загрузчика — nwFlash.

2. По таблице регистров специальных функций (SFR) определите адреса регистров управления и настройки последовательного порта.

3. Определите значение регистров настройки последовательного порта и таймера, используемого для синхронизации.

4. Рассчитайте значения регистров таймера, используемого для синхронизации.

5. Войдите в интегрированную среду программирования Keil-C. Создайте и настройте должным образом проект.

6. Разработайте и введите текст программы в соответствии с созданным алгоритмом.

7. Оттранслируйте программу, и исправьте синтаксические ошибки.

8. Настройте скорость UART терминала программы nwFlash соответственно заданию.

9. Загрузите полученный *.hex файл в лабораторный стенд LESO1.

10. Убедитесь, что в окне терминала вывелась фамилия студента.

4.2 Управление микроконтроллером через последовательный порт (дополнительно)

1. Измените программу таким образом, чтобы данные из микроконтроллера отсылались только по команде, переданной с компьютера. Передача команды осуществляется через терминал nwFlash.

2. Загрузите полученный *.hex файл в лабораторный стенд LESO1.

3. Убедитесь, что программа работает должным образом.

Таблица 5 – Варианты заданий

номер вариант	**скорость UART**
1	300 бит/с
2	600 бит/с
3	1200 бит/с
4	1800 бит/с
5	2400 бит/с
6	4800 бит/с
7	7200 бит/с
8	9600 бит/с
9	600 бит/с

10	1200 бит/с
11	1800 бит/с
12	2400 бит/с
13	4800 бит/с
14	7200 бит/с
15	9600 бит/с

5 Указания к составлению отчета

Отчет должен содержать:

1. Цель работы.
2. Диаграмму передачи данных по последовательному порту.
3. Расчет параметров синхронизации (настройки таймеров).
4. Графическую схему алгоритма работы программы.
5. Исходный текст программы.
6. Содержимое файла листинга программного проекта.
7. Выводы по выполненной лабораторной работе.

Схемы, а также отчет в целом, выполняются согласно нормам ЕСКД.

ТЕСТОВЫЕ ЗАДАНИЯ

1.Микроконтроллер предназначен для

A. построения ПК;

B. радиопередатчиков;

C.систем управления;

D. сопроцессоров;

2.Что нехарактерно для МК

A. наличие встроенных модулей

B. модульный принцип построения;

C.большой объем ОЗУ;

D. наличие регистров общего назначения;

3.Какие параметры характерны современным МП?

A. большой объем встроенной оперативной памяти (ОЗУ)

B. сверхвысокое быстродействие;

C. большие габаритные размеры;

D. наличие встроенных модулей;

4.Чем объясняется коммерческий успех 8-рязрядных МК?

A. широкой периферией;

B. низкой стоимостью;

C. низким энергопотреблением;

D. все ответы правильные;

5.Регистры общего назначения предназначены для

A. долговременного хранения данных

B. хранения программ;

C. оперативного хранения данных;

D. хранения программ и данных;

6.На какие характеристики процессора влияет разделение памяти МК на память программ и память данных?

A. объем памяти

B. энергопотребление;

C. производительность;

D. стоимость;

7.Подсистема памяти микроконтроллера не содержит

A. памяти данных

B. памяти программ;

C. оперативной памяти;

D. ассоциативной памяти;

8.Стек – эта область

A. оперативной памяти микропроцессора

B. постоянной памяти;

C. перепрограммируемой памяти;

D. ассоциативной памяти;

9.Стек предназначен для

A. хранения данных

B. хранения команд;

C. хранения адресов;

D. хранения данных и адресов;

10.Программная модель процессора - это

A. все программно доступные регистры процессора

B. только регистры общего назначения;

C. только указатели памяти;

D. только стек;

11.Периферийные модули микроконтроллера: параллельные порты, АЦП, ЦАП, последовательные интерфейсы, подсистема памяти, драйвер ЖКИ, аналоговый компаратор. Что еще?

A. регистры общего назначения

B. стек;

C. таймеры/Счетчики (Т/С);

D. источники питания;

12.Последовательный интерфейс SPI не используется для

A. программирования МК

B. приема-передачи данных;

C. подключения ЖКИ;

D. тактирования микроконтроллера;

13.Как производится считывание состояния выводов параллельного порта А микроконтроллера?

A. по команде In r16,PortA

B. по команде Mov r16,PinA;

C. по команде In r16,PinA;

D. по команде In r16,DDRA;

14.Как производится запись данных в параллельный порт А микроконтроллера?

A. по команде Out r16,PortA

B. по команде Out potrA,r16;

C. по команде Mov PinA, r16;

D. по команде Out r16,DDRA;

15.Как производится считывание данных с параллельного порта D микроконтроллера?

A. по команде In r16,PortD

B. по команде In r16,PinD;

C. по команде Mov r16,PinD;

D. по команде In r16,DDRD;

16.Как производится запись данных в регистр данных UDR1 UART порта микроконтроллера?

A. по команде Out r16, UDR1

B. по команде Out UDR1,r16;

C. по команде Mov UDR1, r16;

D. по команде Mov r16,UDR1;

17.Как изменяется содержимое указателя стека после выполнения команды CALL?

A. декрементируется на единицу

B. не меняется;

C. декрементируется на две единицы;

D. обнуляется;

18.Как изменяется содержимое указателя стека после выполнения команды RETI?

A. декрементируется на единицу

B. инкрементируется на две единицы;

C. инкрементируется на единицу;

D. обнуляется;

19.На какой режим нельзя настроить Т/С?

A. режим захвата

B. режим сравнения;

C. режим счета реального времени;

D. режим приема-передачи;

20.Подсистема ввода аналоговых сигналов предназначен для

A. суммирования аналоговых сигналов

B. преобразования аналоговых сигналов в частоту;

C. преобразования аналоговых сигналов в цифровой код;

D. преобразования и суммирования аналоговых сигналов;

21.Время преобразование АЦП определяется

A. только частотой генератора тактовых импульсов АЦП

B. только разрядностью АЦП;

C. частотой генератора тактовых импульсов АЦП и разрядностью АЦП;

D. амплитудой входного аналогового сигнала;

22.Последовательный двухпроводной интерфейс IIC (TWI) не используется для

A. программирования МК

B. приема-передачи данных;

C. подключения ЖКИ;

D. подключения АЦП;

23.Скорость обмена по последовательному двухпроводному интерфейсу IIC (TWI) задается

A. значением регистра скорости TWBR ведущего

B. условием «Старт»;

C. условием «Стоп»;

D. адресом ведомого;

24. Каково основное отличие микропроцессора, как универсальной БИС, от других БИС?

A. тип корпуса

B. технология изготовления

C. число выводов микросхемы

D. программируемая логика

25. Какая составная часть микропроцессора выполняет логические и арифметические операции?

A. УУ

B. АЛУ

C. Аккумулятор

D. РОНы

26. Какая архитектура микропроцессора характеризуется разделением памяти данных и памяти программ?

A. Фон-Неймана

B. Гарвардская

C. Принстонская

D. Кембриджская

27. Что не входит в состав микропроцессора?

A. УУ

B. АЛУ

C. ОЗУ

D. РОНы

28. Из перечисленных преимуществ применения микроЭВМ вместо жёстких логических схем выберите неверный ответ(-ы)

A. расширение функций

B. гибкость

C. гораздо быстрее

D. упрощение эксплуатации

E. надёжность

29. Какая из приведенных ниже МОП-технологий наиболее простая и дешевая, но отличается низким быстродействием?

A. P-МОП

B. N-МОП

C. КМОП

D. БиКМОП

30. Какая из приведенных ниже МОП-технологий отличается наибольшим быстродействием?

A. P-МОП

B. N-МОП

C. КМОП

D. БиКМОП

31. Микроконтроллер, выполненный по КМОП технологии, при тактовой частоте, равной 1МГц, имеет ток потребления, равный 2ма, какой будет ток потребления , если увеличить тактовую частоту до 10МГц?

A. 2ма

B. 0,2ма

C. 20ма

D. 10ма

32. Операции какого типа отрабатываются быстрее всего?

A. регистр-регистр

B. регистр-память

C. умножение-деление

D. память-память

33. В каких единицах измеряется скорость выполнения команд?

A. MIPS

B. BAUD

C. MFLOPS

D. MHz

34. Какая схема показывает логические связи между элементами системы и вместе с временными диаграммами полностью определяет функции аппаратной части и даёт полное представление о её работе.

A. структурная

B. принципиальная

C. функциональная

D. потоковая

35. Как называется совокупность важных с точки зрения программиста характеристик микропроцессора (разрядность,система команд ,режимы адресации памяти)

A. структура МП

B. архитектура МП

C. функции МП

D. интерфейс МП

36. В каком регистре запоминается информация о результате выполнения последней обработанной логической или арифметической команды?

A. аккумулятор

B. регистр состояния

C. регистр команд

D. буферный регистр

37. Какой из флагов является признаком переноса из старшего разряда результата?

A. C

B. Z

C. AC

D. N

38. Куда будет помещен результат операции ADD A,R5?

A. в аккумулятор

B. в R5

C. в регистр состояния

D. в ячейку памяти

39. Какой регистр содержит адрес следующей команды, подлежащей выполнению?

A. SP

B. PC

C. PSW

D. A

40. Какой тип стека используется для хранения адресов возврата в микропроцессорах

A. LIFO

B. FIFO

C. MISO

D. MOSI

41. Какой сигнал управления используется при обращении к медленным устройствам (в циклах записи или чтения)?

A. INTA

B. INT

C. I/OR

D. RESET

E. READY

42. Как обычно обозначается у входной сигнал запроса на прерывание

A. RD

B. WR

C. HOLD

D. INT

43. К какому типу относится команда CLR A?

A. безадресная

B. одноадресная

C. двухадресная

44. Что, как правило, указывается в адресной части команды условного перехода

A. абсолютный адрес перехода

B. абсолютный адрес перехода в текущей странице

C. смещение относительно счётчика команд

45. К какому способу адресации относится сегментация памяти?

A. страничная

B. относительная

C. абсолютная

D. индексная

46. Микропроцессоры какой архитектуры быстрее выполняют программу (при одинаковой тактовой частоте)?

A. RISC

B. CISC

C. Гарвардской

D. Фон-неймановской

47. Как называется ввод-вывод, когда нет специальных стробов записи/чтения регистров внешних устройств

A. изолированный

B. отображенный на память

C. потоковый

D. непосредственный

48. В какой момент сбрасывается флажок готовности READY для устройства ввода (например, клавиатуры)

A. при чтении регистра состояния устройства ввода

B. при чтении регистра данных устройства ввода

C. при нажатии клавиши

D. при отпускании клавиши

49. Какой режим ввода-вывода даёт возможность осуществить наиболее быстрый обмен данными между внешними устойствами и оперативной памятью?

A. программный

B. программный форсированный

C. ПДП

D. DMA

E. по прерыванию

50. В каких единицах измеряется скорость передачи данных?

A. бод

B. бит в сек

C. МГц

D. bps

51. Что используется в качестве запоминающего элемента ячейки динамической памяти?

A. триггер на транзисторах

B. входная ёмкость МОП-транзистора

C. выходная ёмкость МОП-транзистора

52. Какой аббревиатурой обозначаются постоянные ЗУ с ультрафиолетовым стиранием?

A. ROM

B. EPROM

C. EEPROM

D. OTPROM

53. По числу больших интегральных схем (БИС) в микропроцессорном комплекте различают микропроцессоры:

A. одноканальные, многоканальные и многоканальные секционные;

B. одноадресные, многоадресные и многоадресные секционные;

C. однокристальные, многокристальные и многокристальные секционные;

D. одноразрядные, многоразрядные и многоразрядные секционные.

54. Система команд, типы обрабатываемых данных, режимы адресации и принципы работы микропроцессора – это:

A. Макроархитектура;

B. Микроархитектура;

C. Миниархитектура;

D. Моноархитектура.

55. С помощью чего микропроцессор координирует работу всех устройств цифровой системы?

A. с помощью шины данных;

B. с помощью шины адреса;

C. с помощью шины управления;

D. с помощью постоянного запоминающего устройства (ПЗУ).

56. Что называется Вводом/выводом (ВВ)?

A. передача данных между ядром ЭВМ, включающим в себя микропроцессор и основную память, и внешними устройствами (ВУ);

B. разрядностью, т.е. максимальным числом одновременно обрабатываемых двоичных разрядов;

C. адреса ячейки памяти, в которой находится окончательный исполнительный адрес;

E. поле памяти с упорядоченной последовательностью записи и выборки информации.

57. Что является структурным элементом формата любой команды?

A. Регистр;

B. Адрес ячейки;

C. Операнд;

E. Код операции (КОП).

58. …….- это процедура или схема преобразования информации об операнде в его исполнительный адрес.

A. Режим кодирования памяти;

B. Режим адресации памяти;

C. Режим формата памяти;

E. Режим обслуживания памяти.

59. Одним из способов обмена памяти к внешним устройствам является:

A. Режим прямого доступа к памяти;

B. Режим формирования сигналов прерываний в памяти;

C. Режим программного управления памятью;

E. Режим обслуживания памяти.

60. Команды распределяют: по функциональному назначению, передача данных, обработка данных, передача управления и …..

A. без адресное;

B. одноадресное;

C. дополнительное;

E. двухадресное.

61. ….- микропроцессоры, в которых начало и конец выполнения операций задаются устройством управления.

A. Универсальные микропроцессоры;

B. Цифровые микропроцессоры;

C. Асинхронные микропроцессоры;

E. Синхронные микропроцессоры.

62. …. - могут быть применены для решения широкого круга разнообразных задач (их эффективная производительность слабо зависит от проблемной специфики решаемых задач)

A. Универсальные микропроцессоры;

B. Цифровые микропроцессоры;

C. Асинхронные микропроцессоры;

E. Синхронные микропроцессоры.

63. …. - различные микроконтроллеры, ориентированные на выполнение сложных последовательностей логических операций, математические МП, предназначенные для повышения производительности при выполнении арифметических операций за счет, например, матричных методов их выполнения.

A. Универсальные микропроцессоры;

B. Синхронные микропроцессоры;

C. Цифровые микропроцессоры;

E. Специализированные микропроцессоры.

64. ….. - это обрабатывающее и управляющее устройство, выполненное с использованием технологии БИС и обладающее способностью выполнять под программным управлением обработку информации, включая ввод и вывод информации, арифметические и логические операции и принятие решений.

A. Процессор;

B. Микропроцессор;

C. Контроллер;

E. Микроконтроллер.

65. …… - это микропроцессорное устройство ориентированное не на производство вычислений, а на реализацию заданной функции управления.

A. Мини-ЭВМ;

B. Микро-ЭВМ;

C. Контроллер;

E. Микроконтроллер.

66. По какой шине передаются лишь выходные сигналы микропроцессора?

A. Шина управления;

B. Шина данных;

C. Шина адреса;

E. Здесь нет нужной шины.

67. Что является важной характеристикой команды?

A. Формат;

B. Процесс;

C. Функциональное назначение;

E. Адрес.

68. Какой из одной букв обозначается разрядность МП?

A. m;

B. a;

C. r;

E. Z.

69. …. это вычислительная или управляющая система выполненная на основе одного или нескольких МП содержащая БИС постоянной и оперативной памяти, БИС управления вводом и выводом информации и оснащенная необходимым периферийным оборудованием (дисплей, печатающее устройство, накопители на магнитных дисках и т. п.).

A. Универсальные - ЭВМ;

B. Мини-ЭВМ;

C. Цифровые – ЭВМ;

E. Микро-ЭВМ.

70. Что означает БУПРПР?

A. База управления последовательности работы программы реестра;

B. Блок управления порядковой работы программы регистра;

C. Блок управлением прерыванием работы процессора;

E. База управлением прерывания работы регистра.

71. Что означает БЗП?

A. Блок защиты памяти;

B. База защиты прерывания;

C. Блок защиты процессора;

E. База защиты процессора.

72. Что означает БС?

A. Блок синхронизации;

B. База синхронизации;

C. Верно и А и Б;

E. Здесь нет правильных ответов.

73. Что означает БУФКА?

A. Блок управления форматированием кода адреса;

B. Блок управление формата кода адресов;

C. База управления форматированием контроллером адреса;

E. Блок управления формированием кодов адресов.

74. Что означает БУВВ?

A. Блок управления выполнением вводом;

B. Блок управления ввода/вывода

C. Блок управления виртуального ввода;

E. Блок управления виртуального вывода;

75. Что означает БУПК?

A. Блок управления последовательности команд;

B. Блок управления прерывания контроллера

C. Блок управления процессора команд;

E. Блок управления памяти команд.

76. Что означает БУВО?

A. Блок управления вводом операции;

B. Блок управления выводом операции;

C. Блок управления виртуальной операции;

E. Блок управления выполнением операции.

77. Чем характеризуется МП?

A. Режимом кодирования памяти;

B. Вводом\Выводом;

C. Тактовой частотой, Разрядностью.

E. Логическим управлением.

78. В общем случае под Архитектурой ЭВМ понимается ….

A. абстрактное представление машины в терминах основных функциональных модулей языка ЭВМ, структуры данных;

B. микропроцессоры включающие в себя систему команд во времени, наличии дополнительных устройств в составе микропроцессора принципы и режимы ЭВМ;

C. только одна программа;

E. абстрактные операции ЭВМ которые имеют одинаковый интерфейс и подключены к единой информационной магистрали.

79. В микропроцессорах используют два метода выработки совокупности функциональных управляющих сигналов:

A. однокристальный и многокристальный;

B. функциональный и тактовый;

C. программный и микропрограммный;

E. универсальный и цифровой.

80. За счёт чего можно расширить операционные возможности микропроцессора ?

A. за счет увеличения числа ПЗУ;

B. за счет увеличения числа памяти данных;

C. за счет увеличения числа регистров;

E. за счет увеличения числа сигналов.

81. Что означает PrСОЗУ?

A. различные секционные многокристальные запоминающие устройства;

B. регистровое сверхоперативное запоминающие устройства;

C. различные сверхоперативное звуковые устройства;

E. реестровое сверхоперативное запоминающие устройства.

82. Что является важнейшим структурным элементом формата любой команды?

A. КОП;

B. Операнд;

C. адрес ячейки;

E. Регистр.

83. Укажите самые распространенные компании, которые занимаются производством микроконтроллеров:

A. Microchip;

B. PIC;

C. Atmel;

D. AVR;

E. Intel;

F. Philips;

G. Scinex;

H. Zilog;

84. Микроконтроллеры делятся на:

A. CISC – устройства;

B. RISC – устройства;

C. DSP – устройства;

D. MIPS – устройства;

85. Производительность микроконтроллера измеряют:

A. в MIPS;

B. в DSP;

C. разрядностью памяти данных;

D. разрядностью памяти программ;

86. Микроконтроллеры по способу программирования классифицируют на:

A. масочно-программируемые;

B. однократно программируемые;

C. перепрограммируемые;

D. флеш-программируемые;

E. последовательно-программируемые;

87. Укажите какие существуют подсемейства для микроконтроллером AVR:

A. tiny;

B. Classic;

C. mega;

D. normal;

E. standart;

88. В микроконтроллерах AVR обозначение EEPROM означает:

A. энергонезависимая память данных;

B. энергонезависимая память программ;

C. регистровая память;

E. сторожевой таймер;

89. Память программ микроконтроллеров семейства AVR разделена на следующие области:

A. область прикладной программы;

B. область загрузчика;

C. область счётчика команд;

D. область энергонезависимой EEPROM;

E. область регистров ввода-вывода;

90. Регистровая память микроконтроллеров семейства AVR включает:

A. 32 регистра общего назначения;

B. 64 регистра общего назначения;

C. область дополнительных регистров ввода-вывода;

D. регистры статического ОЗУ;

91. Выберите правильное утверждение:

A. последние 6 регистров общего назначения объединены в 3 шестнадцатибитных регистра;

B. последние 6 регистров общего назначения объединены в 3 тридцатидвухбитных регистра;

C. последние 8 регистров общего назначения объединены в 4 шестнадцатибитных регистра;

D. последние 8 регистров общего назначения объединены в 4 тридцатидвухбитных регистра;

92. Пусть все выводы PB0…PB7 микроконтроллера ATmega16x/32x используются в качестве входов. К ним подключены кнопки, которые другими выводами подключены к шине питания +5В. Что будет находиться в регистре PinB, когда все кнопки нажаты? Что в этом случае должен

содержать регистр DDRB? Что будет находиться в регистре PinB, когда нажаты все кнопки, кроме кнопки, подключённой к выводу PB7? Выберите правильные утверждения.

A. в регистре PinB будет находится число 0b11111111;

B. в регистре PinB будет находится число 0b00000000;

C. регистр DDRB будет содержать число 0b00000000;

D. регистр DDRB будет содержать число 0b11111111;

E. если все кнопки нажаты кроме кнопки, подключённой к выводу PB7, то в регистре PinB в данном случае будет находиться число 0b01111111;

F. если все кнопки нажаты кроме кнопки, подключённой к выводу PB7, то в регистре PinB в данном случае будет находиться число 0b10000000;

93. Пусть все выводы PB0…PB7 микроконтроллера ATmega16x/32x используются в качестве выходов и подключены к светодиодам. Другие выводы светодиодов подключены через резисторы к общему проводу. Что должен содержать регистр PortB, чтобы все светодиоды были включены? Что в этом случае должен содержать регистр DDRB? Что должен содержать регистр PortB, чтобы были включены все светодиоды, кроме двух центральных? Выберите правильные утверждения:

A. чтобы все светодиоды были включены, регистр PortB должен содержать число 0b11111111;

B. чтобы все светодиоды были включены, регистр PortB должен содержать число 0b00000000;

C. регистр DDRB будет содержать число 0b11111111;

D. регистр DDRB будет содержать число 0b00000000;

E. чтобы были включены все светодиоды, кроме двух центральных регистр PortB должен содержать число 0b11100111;

F. чтобы были включены все светодиоды, кроме двух центральных регистр PortB должен содержать число 0b00011000;

G. содержимое регистра PortB не влияет на включение и выключение светодиодов в данном случае;

94. Выберите правильные утверждения:

A. регистр SREG содержит набор флагов, показывающих текущее состояние микроконтроллера;

B. регистр SREG используется для подключения внешнего ОЗУ;

C. регистр SREG содержит адрес пересылаемого байта по интерфейсу SPI;

D. регистр SREG хранит значение глобальных переменных;

95. Прямая адресация для доступа к данным в микроконтроллерах AVR семейства mega делится на:

A. прямая адресация одного РОН;

B. прямая адресация двух РОН;

C. прямая адресация РВВ;

D. прямая адресация ОЗУ;

E. прямая адресация с индексным регистром;

F. прямая косвенная адресация;

96. Укажите, какой способ адресации изображён на рисунке (см. рис. 1.1):

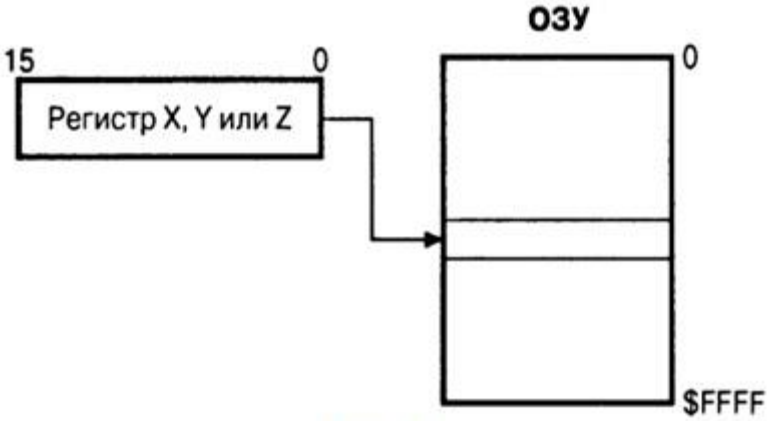

Рис. 1.1.

Рис. 1.1. Какой способ адресации изображён?

A. простая косвенная адресация;

B. прямая адресация одного регистра общего назначения;

C. прямая адресация трёх регистров общего назначения;

D. прямая адресация ОЗУ;

E. относительная косвенная адресация;

97. Укажите, какой способ адресации изображён на рисунке (см. рис. 1.2):

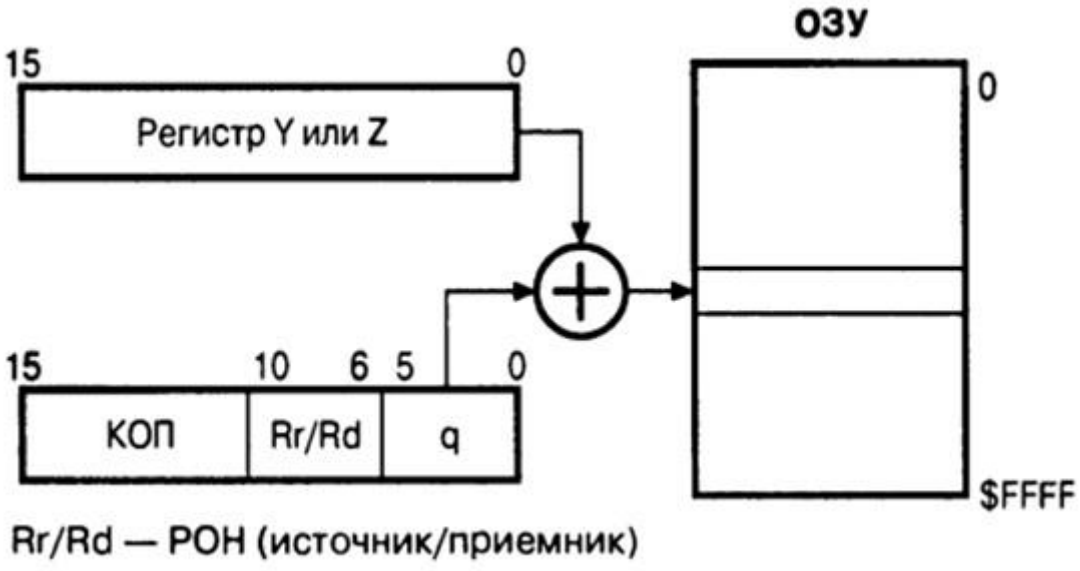

Рис. 1.2.

Рис. 1.2. Какой способ адресации изображён?

A. относительная косвенная адресация;

B. простая косвенная адресация;

C. прямая адресация одного регистра общего назначения;

D. прямая адресация трёх регистров общего назначения;

E. прямая адресация ОЗУ;

98. Для работы с EEPROM-памятью используются регистры:

A. EEAR;

B. EEDR;

C. EECR;

D. EEIR;

E. EEPR;

99. Процедура записи одного байта в EEPROM-память состоит из следующих этапов (Укажите последовательность действий, см. рис. 1.3):

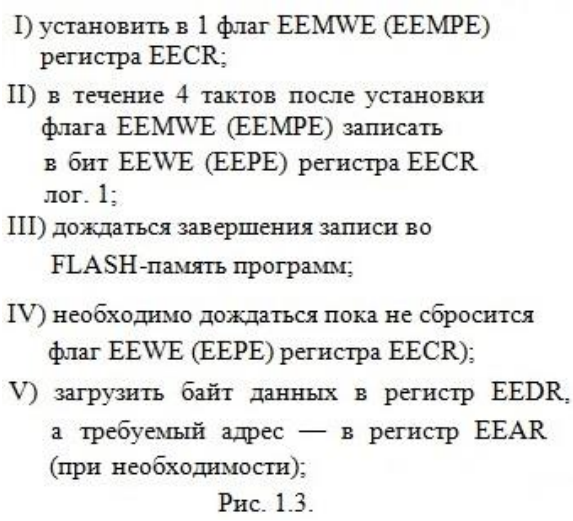

Рис. 1.3. Из каких этапов состоит процедура записи одного байта в EEPROM-память?

A. II-III-I-V-IV;

B. I-V-IV-III-II;

C. IV-III-V-I-II;

D. III-I-V-IV-II;

E. V-I-III-IV-II;

F. I-II-III-IV-V;

G. II-IV-V-III-I;

H. IV-V-I-II-III;

100. Для предотвращения проблем, которые могут возникнуть при записи данных в EEPROM рекомендуется:

A. запрещать все прерывания при выполнении записи в EEPROM; +

B. запрещать все прерывания при выполнении чтения из EEPROM;

C. удерживать микроконтроллер в «спящем» режиме пока производится запись;

D.не знаю…

ЛИТЕРАТУРА

1. Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассеблер.–Спб.: БХВ-Петербург, 2012. Стр 283
2. Алехин, В.А. Микроконтроллеры PIC: основы программирования и моделирования в интерактивных средах MPLAB IDE, mikroC, TINA, Proteus. Практикум / В.А. Алехин. - М.: ГЛТ , 2016. - 248 с.
3. Белов, А.В. Программирование микроконтроллеров для начинающих и не только / А.В. Белов. - СПб.: Наука и техника, 2016. - 352 с.
4. Белов, А.В. Микроконтроллеры AVR: от азов программирования до создания практических устройств / А.В. Белов. - СПб.: Наука и техника, 2016. - 544 с.
5. Бич, М. Микроконтроллеры семейства XC166. Вводный курс разработчика / М. Бич. - М.: ДМК, 2016. - 200 с.
6. Голиков, Д.В. Scratch и Arduino. 18 игровых проектов для юных программистов микроконтроллеров / Д.В. Голиков. - СПб.: BHV, 2018. - 160 с.
7. Евстифеев, А.В. Микроконтроллеры AVR семейства Classic фирмы ATMEL / А.В. Евстифеев. - М.: ДМК, 2015. - 286 с.
8. Евстифеев, А.В. Микроконтроллеры AVR семейства Tiny фирмы ATMEL. Руководство пользователя / А.В. Евстифеев. - М.: ДМК, 2015. - 426 с.
9. Евстифеев, А.В. Микроконтроллеры AVR семейств Mega. Руководство пользователя / А.В. Евстифеев. - М.: ДМК, 2015. - 588 с.
10. Заец, Н.И. Радиолюбительские конструкции на PIC-микроконтроллерах. Кн. 1 / Н.И. Заец. - СПб.: КОРОНА-Век, 2015. - 304 с.
11. Заец, Н.И. Радиолюбительские конструкции на PIC-микроконтроллерах. Книга 4 / Н.И. Заец. - СПб.: Корона-Век, 2015. - 336 с.
12. Заец, Н.И. Радиолюбительские конструкции на Pic-микроконтроллерах кн.1 / Н.И. Заец. - СПб.: Корона-Век, 2015. - 394 с.
13. Иванов, В.Б. Программирование микроконтроллеров для начинающих Визуальное проектирование, язык С, ассемблер / В.Б. Иванов. - СПб.: Корона-Век, 2015. - 176 с.
14. Иванов, В.Б. Программирование микроконтроллеров для начинающих. Визуальное проектирование, язык С, ассемблер / В.Б. Иванов. - СПб.: КОРОНА-Век, 2015. - 176 с.
15. Магда, Ю.С. Микроконтроллеры PIC24. Архитектура и программирование / Ю.С. Магда. - М.: ДМК, 2016. - 240 с.