*МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ*

*Национальный технический университет Украины  
"Киевский политехнический институт"*

*Факультет информатики и вычислительной техники*

*Кафедра вычислительной техники*

***КУРСОВОЙ ПРОЕКТ***

*по дисциплине "Архитектура компьютера".*

*Руководитель работы: Исполнитель:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Ткаченко В. В.) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Пустовит М.А.)*

*Допущен к защите*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Защищен с оценкой Зачетная книжка №7314*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Киев - 2010*

Оглавление

[1 Описание микроконтроллера AVR 4](#_Toc263076391)

[1.1 Архитектура AVR контроллеров 4](#_Toc263076392)

[1.2 Особенности ATmega8515 4](#_Toc263076393)

[1.2.1 Условное графическое обозначение 4](#_Toc263076394)

[1.2.2 Структура микроконтроллера 6](#_Toc263076395)

[1.2.3 Регистры общего назначения 7](#_Toc263076396)

[1.2.4 Распределение памяти данных 7](#_Toc263076397)

[1.2.4.1 Регистр слова состояния SREG 10](#_Toc263076398)

[1.2.4.2 Порты микроконтроллера 11](#_Toc263076399)

[1.2.5 Прерывания 11](#_Toc263076400)

[1.2.6 Таймеры и счётчики 13](#_Toc263076401)

[1.2.7 8-битый таймер 15](#_Toc263076402)

[1.2.8 16-битный таймер/счётчик 16](#_Toc263076403)

[1.3 Система команд AVR 17](#_Toc263076404)

[1.3.1 Команды передачи управления 18](#_Toc263076405)

[1.3.2 Команды работы с битами 19](#_Toc263076406)

[1.3.3 Команды пересылок 19](#_Toc263076407)

[2 Учебно-отладочный стенд EV8031/AVR 27](#_Toc263076408)

[2.1 Технические характеристики. 27](#_Toc263076409)

[2.2 Описание учебно-отладочного стенда 28](#_Toc263076410)

[2.2.1 Работа стенда при загрузке отлаживаемой программы. 28](#_Toc263076411)

[2.2.2 Распределение памяти стенда. 29](#_Toc263076412)

[2.2.3 Структурная схема стенда 30](#_Toc263076413)

[2.2.4 Последовательный приемопередатчик. 30](#_Toc263076414)

[2.2.5 Светодиодный индикатор 32](#_Toc263076415)

[2.2.6 Матричная клавиатура 32](#_Toc263076416)

[2.2.7 Включение портов ОЭВМ и EEPROM памяти 33](#_Toc263076417)

[2.2.8 Расположение элементов, назначение разъемов и перемычек 33](#_Toc263076418)

[2.3 Описание плат расширения 33](#_Toc263076419)

[2.3.1 Назначение. 33](#_Toc263076420)

[2.3.2 Цифроаналоговый преобразователь 34](#_Toc263076421)

[2.3.3 Аналого-цифровой преобразователь 34](#_Toc263076422)

[2.3.4 Генераторы 35](#_Toc263076423)

[2.3.5 Вывод дискретной информации 35](#_Toc263076424)

[2.3.6 Плата расширения для систем автоматического управления 36](#_Toc263076425)

[2.4 Конструкция стенда 37](#_Toc263076426)

[2.5 Работа со стендом 37](#_Toc263076427)

[3 Выполнение индивидуального задания 39](#_Toc263076428)

[3.1 Проектирование микропроцессорной системы 39](#_Toc263076429)

[3.1.1 Описание архитектуры МПС 39](#_Toc263076430)

[3.1.2 Память данных (ПД) 39](#_Toc263076431)

[3.1.3 Программируемый периферийный адаптер (ППА) 39](#_Toc263076432)

[3.1.4 Внешние устройства (ВУ) 40](#_Toc263076433)

[3.1.5 Режим прерываний. 40](#_Toc263076434)

[3.1.6 Прямой доступ к памяти (КПДП) 40](#_Toc263076435)

[3.2 Разработка микропрограмного обеспечения 41](#_Toc263076436)

[3.2.1 Разработка вспомогательных процедур 41](#_Toc263076437)

[3.2.1.1 Распаковка 42](#_Toc263076438)

[3.2.1.2 Запаковка 42](#_Toc263076439)

[3.2.2 Алгоритм сложения/вычитания чисел с плавающей запятой 43](#_Toc263076440)

[3.2.2.1 Блок-схема алгоритма 44](#_Toc263076441)

[3.2.2.2 Листинг программы 45](#_Toc263076442)

[3.2.3 Алгоритм умножения чисел с плавающей запятой по III способу 47](#_Toc263076443)

[3.2.3.1 Функциональная схема умножения 47](#_Toc263076444)

[3.2.3.2 Блок-схема алгоритма умножения 48](#_Toc263076445)

[3.2.3.3 Листинг программы 49](#_Toc263076446)

[3.2.4 Алгоритм вычисления основной функции 50](#_Toc263076447)

[3.2.4.1 Блок-схема алгоримта вычисления основной функции 50](#_Toc263076448)

[3.2.4.2 Листинг программы 51](#_Toc263076449)

[Выводы 52](#_Toc263076450)

[Список использованной литературы 53](#_Toc263076451)

# Описание микроконтроллера AVR

## Архитектура AVR контроллеров

AVR — семейство однокристальных 8-битных RISC микроконтроллеров, выпускаемых компанией Atmel. На момент создания отличительной чертой данного семейства являлось размещение flash-памяти программ на одном кристалле с вычислительным ядром.

Микроконтроллеры семейства AVR имеют модифицированную гарвардскую архитектуру. Классическая гарвардская архитектура предполагает раздельное адресное пространство памяти программ и памяти данных (а значит, и раздельное размещение данных и программ). Это приводит к тому, что код программы не может быть интерпретирован как данные и наоборот (в отличие от неймановской архитектуры, где это возможно). Модификация гарвардской архитектуры, применённая в AVR, заключается в том, что хотя память программ и данных разделены и имеют разное адресное пространство, ячейки памяти программ могут быть считаны и использованы как данные. Причиной для такой модификации является следующий факт: размер памяти программ микроконтроллера обычно значительно превышает размер памяти данных. Однако память программ доступна только для чтения и поэтому не может быть использована как обычная память данных (на чтение и запись). Применение же такой модификации позволяет хранить данные, предназначенные только для чтения, в памяти программ и тем самым экономить память данных. Так удобно хранить, например, таблицы констант.

Современные8-битные AVR контроллеры можно разделить на три основных семейства:

• tinyAVR: память программ от 0,5 до 8 Кб, память данных от 32 до 512 байт, небольшое число линий ввода-вывода (4-28), ограниченный набор периферийных устройств.

• megaAVR: память программ от 8 до 256 Кб, память данных от 0,5 до 8 Кб, 23 – 86 линий ввода-вывода, в набор периферийных устройств могут входить АЦП, интерфейс последовательного порта (USART), КПП, интерфейс JTAG, сторожевой таймер WDT.

• XMEGA: память программ от 16 до 384 Кб, память данных от 2 до 32 Кб, от 32 линий ввода-вывода, в набор периферийных устройств входят АЦП, ЦАП, несколько интерфейсов последовательного порта, КПП (с возможностью использования любой линии ввода-вывода как входа прерывания), сторожевой таймер WDT, 4-канальный КПДП.

Набор поддерживаемых команд в этих семействах практически не отличается. Базовыми можно считать приблизительно 130 команд, которые поддерживаются всеми контроллерами. Конкретные модели могут поддерживать дополнительные команды, например в контроллерах XMEGA аппаратно реализованы алгоритмы шифрования DES и AES.

Основные отличия между контроллерами внутри семейства заключаются в размере памяти программ и данных, а так же наборе периферии (например, в ATmega16 есть АЦП, а в ATmega8515 — нет).

## Особенности ATmega8515

### Условное графическое обозначение

На рис. 1.1 изображено УГО ATmega8515. Микроконтроллер выпускается в корпусах PDIP, TQFP, MLF, PLCC. В таблице 2.1 приведены описания выводов контроллера.

Выводы портов PA, …, PE в зависимости от программных настроек контроллера могут работать как порты ввода-вывода общего назначения (в этом случае логические уровни на выводах портов управляются кодом программы), а могут выполнять и альтернативные функции. Описание альтернативных функций приведено в таблице . Подробно работа выводов в альтернативных режимах описана ниже в соответствующих разделах.

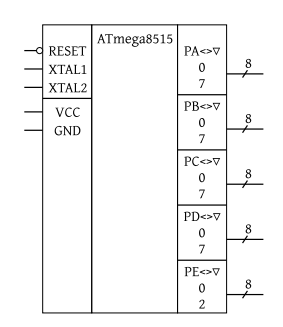


Рисунок . Условное графическое обозначение ATmega8515

Таблица . Описание выводов ATmega8515

|  |  |
| --- | --- |
| **Название** | **Описание** |
|  | Сброс |
| XTAL1, XTAL2 | Входы для подключения внешнего источника тактовой частоты |
| VCC | Питание +4,5…+5,5 В |
| GND | Земля |
| PA0…PA7, PB0…PB7, PC0…PC7, PD0…PD7 | 8-битные двунаправленные порты ввода-вывода с возможностью подключения подтягивающих резисторов |
| PE0…PE2 | 3-битный двунаправленный порт ввода-вывода с возможностью подключения подтягивающих резисторов |

Таблица . Описания альтернативных функций выводов портов ATmega8515

|  |  |
| --- | --- |
| **Вывод порта** | **Альтернативная функция** |
| PA0…PA7 | AD0…AD7: шина адреса и данных для подключения внешней памяти данных |
| PB0 | T0: внешний синхросигнал таймера 0;  OC0: сигнал о совпадении значения таймера 0 с ожидаемым значением |
| PB1 | T1: внешний синхросигнал таймера/счётчика 1 |
| PС0…PС7 | A8…A15: старшая часть шины адреса для подключения внешней памяти данных |
| PD2 | INT0: вход внешнего прерывания 0 |
| PD3 | INT1: вход внешнего прерывания 1 |
| PD5 | OC1A: сигнал о совпадении значения таймера/счётчика 1 с ожидаемым значением А |
| PD6 | : строб записи во внешнюю память данных |
| PD7 | : строб чтения из внешней памяти данных |
| PE0 | INT2: вход внешнего прерывания 2  ICP: внешние событие для таймера/счётчика 1 |
| PE1 | ALE (Address Latch Enable): строб выдачи младшей части адреса |
| PE2 | OC1B: сигнал о совпадении значения таймера/счётчика 1 с ожидаемым значением В |

### Структура микроконтроллера

УпрощённаяструктурнаясхемаATmega8515 приведена на рис. 2.2. На схеме обозначены:

• УУ — устройство управления;

• PC (program counter) — счётчик команд;

• SP (stack pointer) — указатель вершины стека;

• SREG (status register) — регистр слова состояния;

• R0…R31 — регистры общего назначения;

• АЛУ — арифметико-логическое устройство;

• ПП — память программ;

• ПД — память данных;

• КПП — контроллер приоритетных прерываний;

• EEPROM — энергонезависимая память;

• PORTA…PORTE — порты ввода-вывода;

• СШ — системная шина.

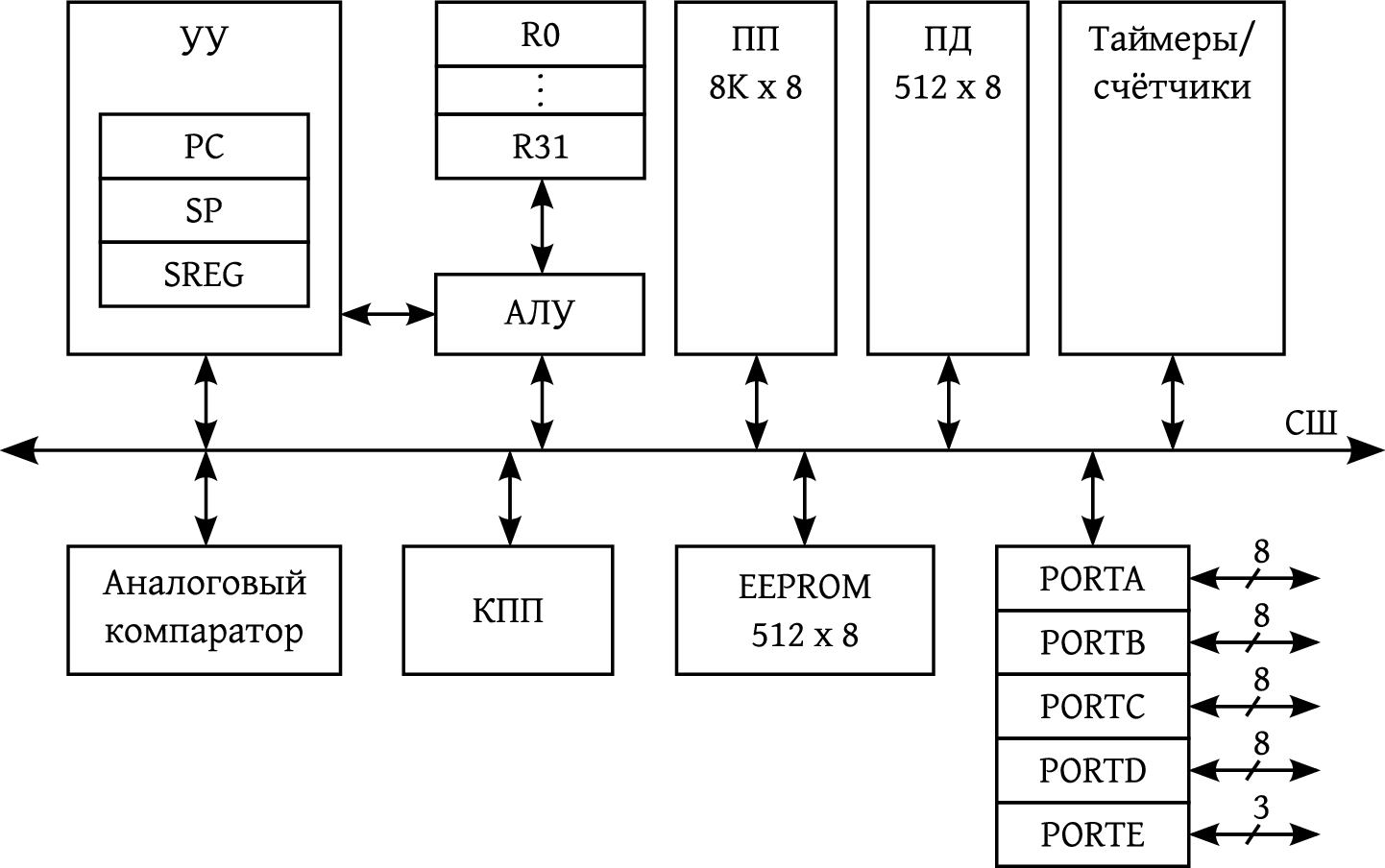


Рисунок . Структурная схема ATmega8515

### Регистры общего назначения

Любой AVR процессор имеет 32 8-битных регистра общего назначения, имеющих названия от R0 до R31. Некоторые команды принимают операнды, состоящие из пары регистров, рассматривая два 8-битных регистра как один 16-битный. Пара регистров обозначается Rr:Rd. Младший из пары регистров содержит младшие биты числа, т. е. число имеет порядок байт little endian.

Старшие 3 пары регистров используются как 16-битные указатели адреса в командах косвенной адресации. Для этих пар назначены специальные имена X, Y, Z (рис. )

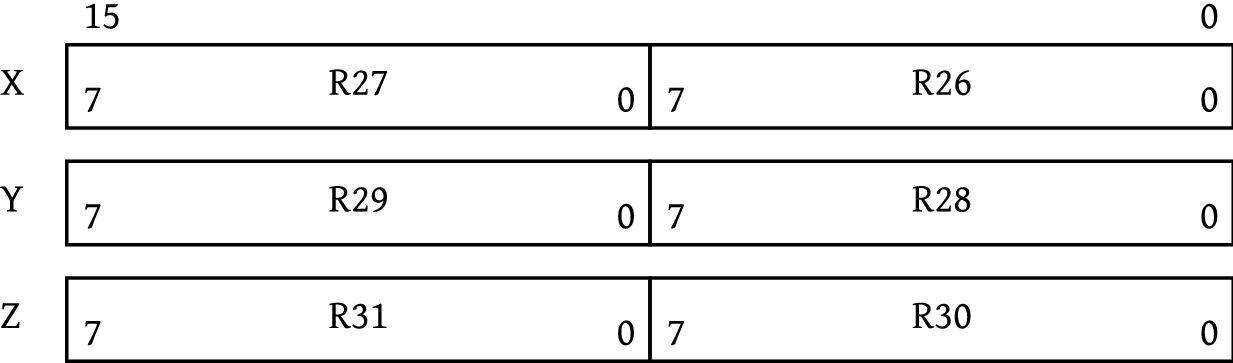


Рисунок . Указатели адреса — регистры X, Y, Z

Регистры общего назначения включены в адресное пространство памяти данных по адресам $00…$1F.

### Распределение памяти данных

В единое адресное пространство памяти данных отображаются регистры общего назначения, служебные регистры, внутренняя и внешняя память данных. Карта распределения адресного пространства памяти данных приведена на рис. . На карте обозначены следующие адреса:

• $0000…$001F — регистры общего назначения R0…R31;

• $0020…$005F — 64 порта ввода/вывода (среди них: регистр слова состояния, указатель стека, регистры таймеров, порты микроконтроллера);

• $0060…$025F — резидентная память данных объёмом 512 x 8;

• $0260…$FFFF — внешняя память данных (если подключена).

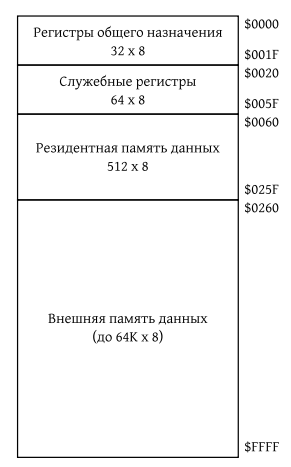


Рисунок . Карта адресного пространства памяти данных

На рис. 2.4 показана схема подключения внешней памяти данных. При подключении памяти данных используются альтернативные функции портов микроконтроллера (табл. 2.2). Следует заметить, что на рисунке показано подключение максимально возможного объёма — 64 Кб. При подключении внешней памяти меньших размеров несколько старших разрядов шины адреса будут не подключены.

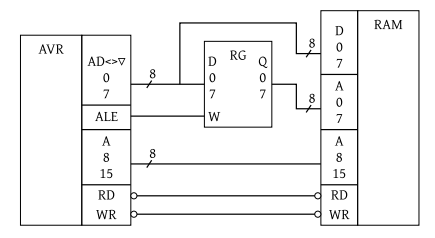


Рисунок . Схема подключения внешней памяти данных

Порты ввода/вывода предназначены для управления режимами работы различных узлов микроконтроллера (порты микроконтроллера, таймеры, контроллер прерываний, настройка внешней памяти данных, АЦП, UART и др.) Всего в микроконтроллере 64 порта ввода/вывода и они имеют номера $00…$3F.

В микроконтроллере предусмотрено два способа работы с портами ввода/вывода. Первый способ предусматривает использование специальных команд IN и OUT. В качестве операндов этим командам передаётся непосредственно номер порта (то есть, число $00…$3F). Кроме того, предусмотрены команды CBI и SBI для побитовой работы с портами ввода/вывода $00…$1F.

Поскольку порты ввода/вывода отображаются в адресное пространство данных, то к ним можно получить доступ и при помощи стандартных команд работы с памятью данных (LD и ST). Чтобы вычислить адрес порта ввода/вывода в памяти, необходимо к адресу порта ввода/вывода необходимо добавить $20 (см. карту распределения памяти данных). Таким образом, обращение к портам ввода-вывода возможно по адресам $20…$5F памяти данных.

**Пример.** Загрузить 0 в порт ввода-вывода SREG двумя способами.

.include "m8515def.inc"

.cseg

; Первый способ, через порты ввода/вывода

ldi r25, 0 ; r25 := 0

out SREG, r25 ; SREG := r25

; Второй способ, через память данных

ldi r25, 0 ; r25 := 0

sts SREG + 0x20, r25 ; SREG := r25

**Важно:** не следует путать порты ввода/вывода и порты микроконтроллера. Порты ввода/вывода управляют настройками периферии (и в том числе портами микроконтроллера). Порты микроконтроллера — физические группы пинов микросхемы.

Описание портов ввода/вывода приведено в таблице 2.3.

Таблица .

|  |  |  |  |
| --- | --- | --- | --- |
| **Порт/Адрес** | | **Название** | **Описание** |
| $05 | $25 | PINE | Состояние порта Е |
| $06 | $26 | DDRE | Направление битов порта Е |
| $07 | $27 | PORTE | Порт ввода/вывода Е |
| $10 | $30 | PIND | Состояние порта D |
| $11 | $31 | DDRD | Направление битов порта D |
| $12 | $32 | PORTD | Порт ввода/вывода D |
| $13 | $33 | PINC | Состояние порта C |
| $14 | $34 | DDRC | Направление битов порта C |
| $15 | $35 | PORTC | Порт ввода/вывода C |
| $16 | $36 | PINB | Состояние порта B |
| $17 | $37 | DDRB | Направление битов порта B |
| $18 | $38 | PORTB | Порт ввода/вывода B |
| $19 | $39 | PINA | Состояние порта A |
| $1A | $3A | DDRA | Направление битов порта A |
| $1B | $3B | PORTA | Порт ввода/вывода A |
| $24 | $44 | ICR1L | Таймер/счётчик 1, младший байт счётчика внешних событий |
| $25 | $45 | ICR1H | Таймер/счётчик 1, старший байт счётчика внешних событий |
| $28 | $48 | OCR1BL | Младший байт ожидаемого значения В таймера 1 |
| $29 | $49 | OCR1BH | Старший байт ожидаемого значения В таймера 1 |
| $2A | $4A | OCR1AL | Младший байт ожидаемого значения A таймера 1 |
| $2B | $4B | OCR1AH | Старший байт ожидаемого значения A таймера 1 |
| $2C | $4C | TCNT1L | Младший байт таймера/счётчика 1 |
| $2D | $4D | TCNT1H | Старший байт таймера/счётчика 1 |
| $2E | $4E | TCCR1B | Регистр B управления таймером/счётчиком 1 |
| $2F | $4F | TCCR1A | Регистр A управления таймером/счётчиком 1 |
| $31 | $51 | OCR0 | Ожидаемое значение таймера/счётчика 0 |
| $32 | $52 | TCNT0 | Таймер/счётчик 0 |
| $33 | $53 | TCCR0 | Регистр управления таймером/счётчиком 0 |
| $35 | $55 | MCUCR | Регистр управления микроконтроллером |
| $36 | $56 | EMCUCR | Расширенный регистр управления микроконтроллером |
| $38 | $58 | TIFR | Регистр флагов прерываний таймеров/счётчиков |
| $39 | $59 | TIMSK | Регистр маски прерываний таймеров/счётчиков |
| $3A | $5A | GIFR | Регистр глобальных флагов прерываний |
| $3B | $5B | GICR | Регистр глобального управления прерываниями |
| $3D | $5D | SPL | Младший байт указателя стека |
| $3E | $5E | SPH | Старший байт указателя стека |
| $3F | $5F | SREG | Регистр слова состояния |

Рассмотрим наиболее важные служебные регистры.

#### Регистр слова состояния SREG

Регистр слова состояния SREG (рис. 2.6) содержит флаги, устанавливаемые согласно результату выполнения последней инструкции, а также флаги управляющие работой микроконтроллера.

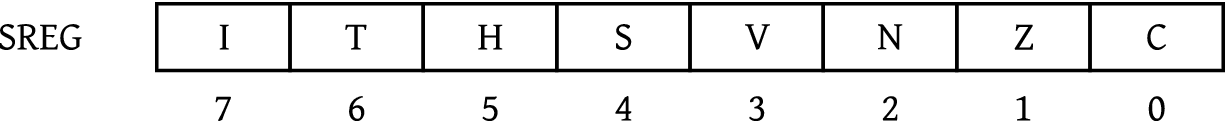


Рисунок . Порт ввода/вывода SREG — регистр слова состояния

Биты порта ввода/вывода SREG имеют следующее назначение:

Бит 0. C — флаг переноса из старшего разряда при выполнении арифметико-логических команд или операций сдвига.

Бит 1. Z — флаг нулевого результата арифметико-логической операции.

Бит 2. N — флаг отрицательного результата арифметико-логической операции. Совпадает со значением старшего бита результата.

Бит 3. V — флаг переполнения, устанавливается в случае перехода с $FF в $00 в ходе последней арифметической операции.

Бит 4. S = N  V — флаг знака. Позволяет проверить, была ли операция выполнена с отдельной обработкой знака или с беззнаковыми операндами.

Бит 5. H — флаг переноса между тетрадами (т. e., из 3-го разряда в 4-й). Используется для выполнения двоично-десятичной коррекции.

Бит 6. T — пользовательский флаг. Не имеет служебного назначения и может использоваться пользователем по своему усмотрению. Изменяется только специальными командами изменения этого флага (BLD, BST).

Бит 7. I — флаг глобального разрешения прерываний. Если флаг установлен — разрешение/запрет прерываний определяется настройками соответствующих узлов микроконтроллера. Если флаг сброшен — все прерывания запрещены вне зависимости от других настроек. Аппаратно сбрасывается при появлении прерывания и устанавливается командой возврата из прерывания.

Вописаниисистемыкомандуказано,какиеименнофлагиизменяютсякаждойкомандой. При переходе на обработчик прерывания регистр слова состояния автоматически не сохраняется, поэтому подпрограмма-обработчик прерывания должна сохранять и восстанавливать его значение.

#### Порты микроконтроллера

Микроконтроллер имеет четыре 8-битных двунаправленных порта (A, B, C, D) и один 3-битный (E). Порты обозначаются PORTx, где x — имя порта (A, B, C, D, E). Каждый порт настраивается тремя портами ввода/вывода, каждый бит которых соответствует одному пину порта микроконтроллера.

• DDRx — порт направления. Для каждого пина порта может быть отдельно задано направление: 0 — ввод, 1 — вывод.

• PORTx — порт данных. В режиме вывода используется для вывода данных. В режиме ввода: 0 — высокоомное состояние, 1 — подключение резистора-подтяжки.

• PINx — порт входных данных. Содержит текущее значение пинов порта, вне зависимости от режима. Предназначен только для чтения.

Чтобыизбежатьвременнойнестабильностиприизменениизначенияпинапортаизвне,накаждомпинеустановленсинхронизирующийтриггер. Из-за этого новое значение порта устанавливается с задержкой 0.5–1.5 машинных цикла. Временные диаграммы приведены.

### Прерывания

В микропроцессор встроен централизованный контроллер приоритетных прерываний. По умолчанию таблица векторов прерываний расположена по младшим адресам памяти программ. Производитель рекомендует по адресу вектора прерывания размещать инструкцию RJMP. Чем больше вектор прерывания — тем ниже приоритет. Описание векторов прерываний приведено в таблице .

Таблица 1.4 Векторы прерываний в ATmega8515

|  |  |  |  |
| --- | --- | --- | --- |
| **Вектор** | **Адрес** | **Название** | **Описание** |
| 1 | $000 | RESET | Сброс |
| 2 | $001 | INT0 | Внешнее прерывание 0 |
| 3 | $002 | INT1 | Внешнее прерывание 1 |
| 4 | $003 | TIMER1 CAPT | Внешнее событие. Увеличивает таймер/счётчик 1 в режиме подсчета внешних событий. |
| 5 | $004 | TIMER1 COMPA | Значение таймера/счётчика 1 совпало с ожидаемым значением А |
| 6 | $005 | TIMER1 COMPB | Значение таймера/счётчика 1 совпало с ожидаемым значением В |
| 7 | $006 | TIMER1 OVF | Переполнение таймера/счётчика 1 |
| 8 | $007 | TIMER0 OVF | Переполнение таймера/счётчика 0 |
| 14 | $00D | INT2 | Внешнее прерывание 2 |
| 15 | $00E | TIMER0 COMP | Значение таймера/счётчика 0 совпало с ожидаемым значением |

Прерывания можно глобально запретить, установив в 0 бит I регистра слова состояния SREG. Данный флаг запрета имеет преимущество перед всеми другими флагами разрешения прерываний.

Внешние прерывания разрешаются установкой флагов INT0, INT1, INT2 порта ввода/вывода GICR в 1 (рис. 2.7).

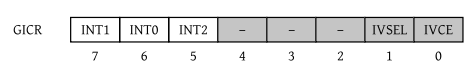


Рисунок . Порт ввода/вывода GICR — регистр глобального управления прерываниями

Для внешних прерываний можно задать вид входного сигнала, при котором будет зафиксирован запрос прерывания. Прерывания INT0 и INT1 имеют 4 режима, которые задаются 2 битами ISCn1:ISCn0 порта ввода/вывода MCUCR, где n — номер внешнего прерывания (рис. 2.8). Возможные комбинации значений приведены в таблице 2.5. Если настроить фиксацию запроса прерывания по низкому уровню сигнала, то запрос прерывания будет фиксироваться до тех пор, пока сигнал не примет высокий уровень.

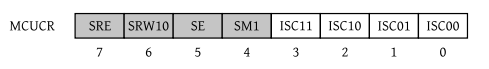


Рисунок . Порт ввода/вывода MCUCR — регистр управления микроконтроллером

Таблица . Режимы фиксации запроса прерываний INT0 и INT1

|  |  |
| --- | --- |
| **ISCn1:ISCn0** | **Запрос прерывания** |
| 00 | Низкий уровень |
| 01 | Любое изменение логического уровня |
| 10 | Отрицательный фронт |
| 11 | Положительный фронт |

Канал запроса прерывания INT2 всегда опрашивается асинхронно, то есть, независимо от тактовых импульсов. Импульс длительностью более может быть использован в качестве запроса прерывания. INT2 имеет только 2 режима фиксации запроса, которые выбираются младшим битом ISC2 порта ввода-вывода EMCUCR (рис. 2.9, таблица 2.6).

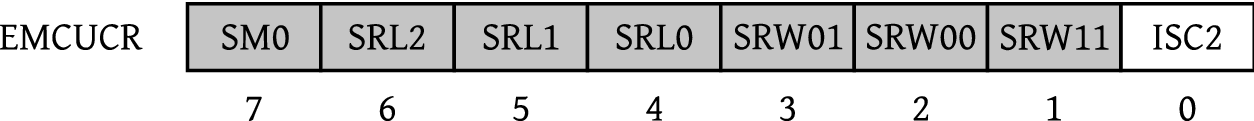


Рисунок . Порт ввода/вывода EMCUCR — расширенный регистр управления микроконтроллером

Таблица . Режимы фиксации запроса прерывания INT2

|  |  |
| --- | --- |
| ISC2 | **Запрос прерывания** |
| 0 | Отрицательный фронт |
| 1 | Положительный фронт |

Перепад сигнала на пинах запроса внешнего прерывания так же фиксируется в служебном регистре GIFR, даже если прерывания были запрещены тем или иным образом. Но поскольку для INT0 и INT1 существует режим, где запросом прерывания является уровень сигнала, а не его перепад, в этом режиме флаги перепада не устанавливаются.

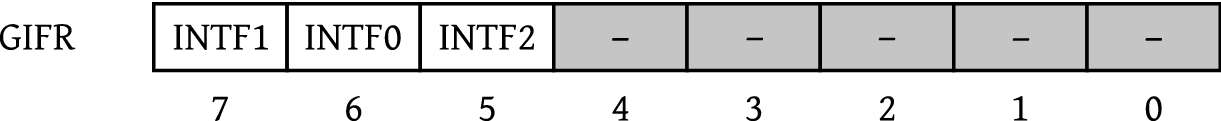


Рисунок . Порт ввода/вывода GIFR — регистр глобальных флагов прерываний

При переходе на обработчик прерывания микроконтроллер автоматически сохраняет только адрес возврата, а SREG не сохраняет. Поэтому если обработчик прерывания изменяет SREG (а большинство обработчиков, выполняющих хоть какие-то осмысленные действия изменяют SREG), то он должен сохранить и восстановить его самостоятельно.

### Таймеры и счётчики

В состав микроконтроллера входят два таймера/счётчика: 8-битый TCNT0 и 16-битный TCNT1. Каждый из таймеров имеет в зависимости от разрядности один или два порта ввода/вывода управления, текущего значения и ожидаемого значения. Оба таймера могут выдавать несколько прерываний в различных режимах работы. Разрешение или запрет этих прерываний производится установкой битов порта ввода/вывода TIMSK (рис. 2.11).

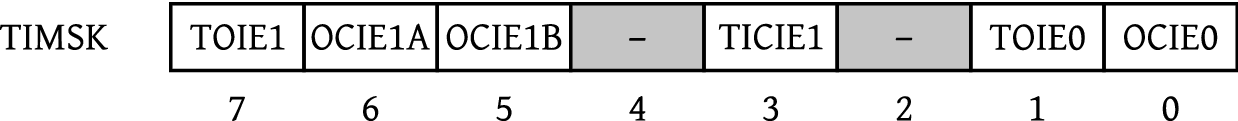


Рисунок . Порт ввода/вывода TIMSK — регистр маски прерываний таймеров/счётчиков

Биты порта ввода/вывода TIMSK имеют следующее назначение:

• OCIE0 — прерывание по совпадению значения таймера/счётчика 0 с ожидаемым значением;

• TOIE0 — прерывание по переполнению таймера/счётчика 0;

• TICIE1 — прерывание при внешнем событии, увеличивающем значение таймера/счётчика 1;

• OCIE1B — прерывание по совпадению значения таймера/счётчика 1 с ожидаемым значением В;

• OCIE1A — прерывание по совпадению значения таймера/счётчика 1 с ожидаемым значением А;

• TOIE1 — прерывание по переполнению таймера/счётчика 1.

Для разрешения прерывания соответствующий бит регистра TIMSK должен быть установлен в 1, а для запрета — сброшен в 0.

Так же оба таймера разделяют регистр флагов таймеров TIFR (рис. 2.12). Флаги таймеров устанавливаются вне зависимости от разрешения соответствующих прерываний. Однако, если прерывания разрешены, то при переходе на обработчик прерывания флаги сбрасываются аппаратно. Флаг может быть сброшен вручную, для этого в соответствующий бит необходимо записать 1.

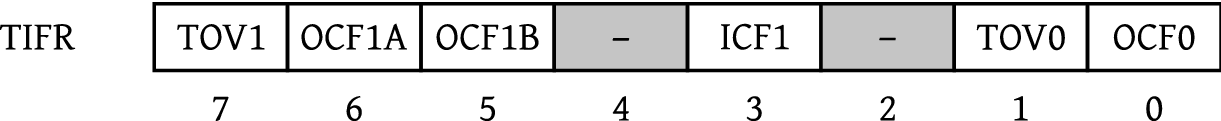


Рисунок . Порт ввода/вывода TIFR — регистр флагов прерываний таймеров/счётчиков

Значения флагов в TIFR:

• OCF0 — флаг совпадения значения таймера/счётчика 0 с ожидаемым значением;

• TOV0 — флаг переполнения таймера/счётчика 0;

• ICF1 — флаг регистрации внешнего события;

• OCF1B — флаг совпадения значения таймера/счётчика 1 с ожидаемым значением В;

• OCF1A — флаг совпадения значения таймера/счётчика 1 с ожидаемым значением А;

• TOV1 — флаг переполнения таймера/счётчика 1.

Оба таймера/счётчика могут управляться как от внутреннего, так и от внешнего источника синхронизирующих импульсов. Внутренние синхроимпульсы проходят через предварительный делитель частоты, который может работать в одном из 5 режимов. Оба таймера/счётчика используют один и тот же делитель частоты, но могут работать в разных режимах. Внешний синхроимпульс опрашивается в ходе каждого машинного цикла и может увеличивать значение таймера по переднему либо заднему фронту сигнала. Логика детектора фронта сигнала и синхронизации вносит задержку в 2.5–3.5 машинных цикла между непосредственно фронтом сигнала и увеличением таймера/счётчика. Для корректной обработки сигналов внешнего источника синхроимпульсов необходимо, чтобы его частота была минимум вдвое меньше тактовой частоты микроконтроллера. Используемый источник синхронизирующих импульсов задаётся битами CSn2:CSn0 порта ввода/вывода TCCRn, где n — номер таймера/счётчика.

Таблица .

|  |  |
| --- | --- |
| **CSn2:CSn0** | **Режим** |
| 000 | Таймер выключен |
| 001 | Внутренний источник с частотой |
| 010 | Внутренний источник с частотой |
| 011 | Внутренний источник с частотой |
| 100 | Внутренний источник с частотой |
| 101 | Внутренний источник с частотой |
| 110 | Внешний источник по отрицательному фронту импульса |
| 111 | Внешний источник по положительному фронту импульса |

### 8-битый таймер

Микроконтроллер содержит 8-битный таймер TCNT0. Таймер может управляться как внутренними синхроимпульсами через предварительный делитель частоты, так и внешним источником. В процессе работы таймер может генерировать прерывания TIMER0 OVF (номер 8) и TIMER0 COMP (номер 15).

Таймер/счётчик содержит широтно-импульсный модулятор и имеет режимы работы, связанные с генерированием импульсов различной длительности на выходных портах, но рассмотрение этих режимов выходит за рамки данного курса. Далее будем рассматривать только стандартный режим.

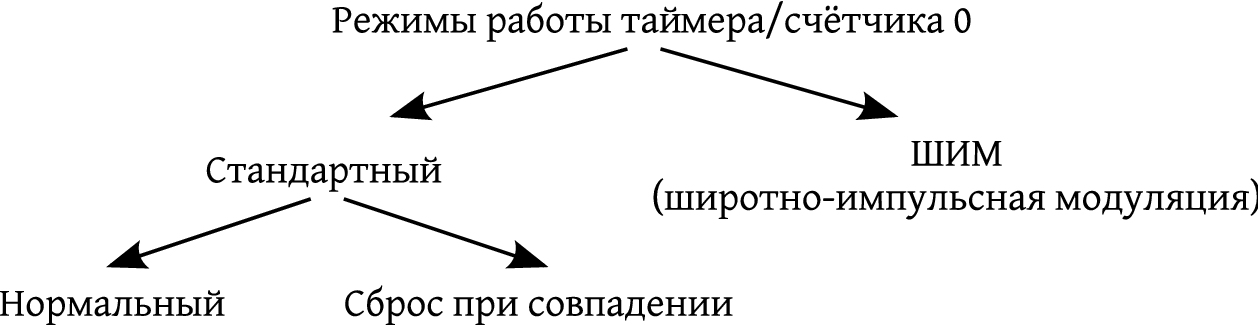


Рисунок . Режимы работы таймера/счётчика 0

Таймер/счётчик TCNT0 имеет 2 режима работы без использования широтно-импульсного модулятора: нормальный и со сбросом таймера при совпадении.

Все параметры таймера/счётчика 0 устанавливаются через порт ввода/вывода TCCR0 (рис. ). Ранее рассмотренные биты выбора синхросигнала CS02:CS00 расположены в этом порту. Остальные биты рассмотрены ниже.

Текущее значение таймера/счётчика 0 хранится в порту ввода/вывода TCNT0. Ожидаемое значение хранится в порту OCR0.

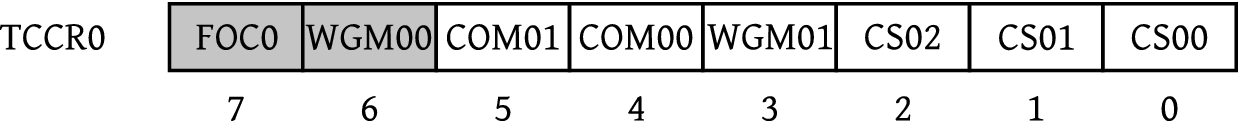


Рисунок . Порт ввода/вывода TCCR0 — регистр управления таймером/счётчиком 0

В нормальном режиме к TCNT0 с указанной частотой или по внешнему сигналу прибавляется единица. При совпадении значения счётчика TCNT0 и ожидаемого OCR0 устанавливается флаг OCF0 регистра TIFR, и, если разрешено, генерируется прерывание TIMER0 COMP.

Режим сброса при совпадении, как следует из названия, отличается от нормального тем, что при совпадении текущего значения с ожидаемым, текущее сбрасывается в 0 (но таймер при этом продолжает работать). Режим работы задаётся битом WGM01 регистра TCCR0 в соответствии с таблицей.

|  |  |  |
| --- | --- | --- |
| **WGM01** | **Режим** | **Диапазон** |
| 0 | Нормальный | 0x00…0xFF |
| 1 | Сброс при совпадении | 0x00…(OCR0) |

В любом режиме запись значения в один из регистров таймера/счётчика имеет больший приоритет и будет выполнена (например, когда счётчик запущен и в одном и том же машинном цикле увеличивается значение таймера и в него записывается константа, то в результате в таймере гарантировано будет находиться значение константы). Также в этом цикле не будет выполнена проверка совпадения, и если счётчик запущен и записываемое значение равно ожидаемому, прерывание по совпадению будет потеряно. Эту особенность следует учитывать при программировании таймера/счётчика, работающего без остановки.

Вмикроконтроллерепредусмотренспециальныйрежимвыдачиуправляющегосигналаприсовпадениитекущегозначениятаймерасожидаемым. Управляющий сигнал может выдаваться только через специальный пин OC0 (альтернативное назначение пина PB0). Для работы в данном режиме пин PB0 должен быть настроен на вывод. Форма управляющего сигнала задаётся битами COM01:COM00 регистра TCCR0.

|  |  |
| --- | --- |
| **COM01:COM00** | **Вид сигнала** |
| 00 | Вывод OC0 отключен, сигнал не выдаётся |
| 01 | Изменяет значение OC0 на противоположное |
| 10 | Выдает логический 0 на OC0 |
| 11 | Выдает логическую 1 на OC0 |

Следует отметить, для выдачи управляющего сигнала используется тот же пин микроконтроллера PB0, что и для внешнего источника синхроимпульсов. Поэтому эти два режима совмещать невозможно.

**Пример.** Настроить таймер: нормальный режим, ожидаемое значение равно 0x42. В цикле ожидать флага совпадения таймера с ожидаемым значением.

### 16-битный таймер/счётчик

Микроконтроллер содержит 16-битый таймер/счётчик, который обладает значительно большими возможностями, чем 8-битый. В частности, он предоставляет 2 независимых регистра ожидаемых значений. Таймер/счётчик в процессе работы может формировать прерывания 4–7, в зависимости от их разрешения и режима работы таймера. Он также содержит более сложный механизм генерации импульсов и широтно-импульсной модуляции, однако его рассмотрение выходит за рамки курса.

Все регистры данных, с которыми оперирует таймер/счётчик являются 16-битными. Для работы с ними в микроконтроллере предусмотрен специальный временный 8-битый регистр, через который происходит работа со старшим байтом. Доступ необходимо выполнять последовательно в два этапа. При этом между ними может быть вызвано прерывание, которое изменит состояние 16-битного регистра и прочитанные данные станут недействительными. Чтобы избежать этого, необходимо блокировать прерывания перед ручным доступом к 16-битным регистрам. К старшему байту таймера/счётчика процессор обращается косвенно, при помощи временного регистра. Причем значение временного регистра синхронизируется со значением старшего байта в момент обращения к младшему байту. Поэтому при записывать первым необходимо старший байт, а считывать младший.

in r18, SREG ; r18

cli ;

ldi r17, 0x12

ldi r16, 0x34

out TCNT1H, r17 ;

out TCNT1L, r16 ;

; --------------

in r14, TCNT1L ;

in r15, TCNT1H ;

out SREG, r18 ;

16-битный таймер/счётчик, без использования широтно-импульсного модулятора, имеет два таких же режима работы, как и 8-битный таймер: нормальный, и со сбросом таймера при совпадении текущего значения с ожидаемым. Приоритет записи процессором значений в регистры счётчика перед внутренним изменением сохраняется.

В нормальном режиме с установленной частотой к значению таймера/счётчика прибавляется единица. Текущее значение таймера/счётчика сравнивается с регистровыми парами OCR1AH:OCR1AL и OCR1BH:OCR1BL вне зависимости от режима работы. В случае совпадения устанавливаются флаги OCF1A и OCF1B регистра TIFR соответственно. Если разрешены, генерируются соответствующие прерывания. Также при этом таймер может выдавать сигнал на пины микроконтроллера OC1A и OC1B. Форма сигнала определяется настройкой битов COM1A1:COM1A0 и COM1B1:COM1B0 соответственно для каждого из пинов. Значения битов для определённой формы сигнала такие же, как и в 8-битном таймере.

Таймер содержит пару регистров ICR1H:ICR1L, которая используется как счётчик внешних событий, зафиксированных соответствующим запросом пина PE.0. Событием может являться отрицательный либо положительный фронт ICP в зависимости от настройки бита ICES1 порта ввода-вывода TCCR1B.

|  |  |
| --- | --- |
| **ICES1** | **Вид сигнала внешнего события** |
| 0 | Отрицательный фронт |
| 1 | Положительный фронт |

При возникновении внешнего события текущее значение таймера копируется в регистровую паруICR1H:ICR1L, устанавливается флаг ICF1 и, если разрешено, генерируется соответствующее прерывание. После чего таймер продолжает работу в соответствии с режимом.

Режим сброса при совпадении может использовать регистровые пары OCR1AH:OCR1AL и ICR1H:ICR1L для сравнения с текущим значением в соответствии с настройкой. Последнее по сути является сбросом таймера при появлении внешнего сигнала и позволяет измерять время между внешними сигналами.

|  |  |  |
| --- | --- | --- |
| **WGM13 WGM12 WGM11** | **Режим** | **Диапазон** |
| 000 | Стандартный | 0x0000…0xFFFF |
| 010 | Сброс по совпадению с OCR1A | 0x0000…OCR1A |
| 110 | Сброс по совпадению с ICR1 | 0x0000…ICR1 |

## Система команд AVR

В описании системы используется нотация операторов языка С. Для обозначения последовательности регистров, содержащих части одного числа используется символ “:”. А также приняты следующие обозначения:

Rd — регистр назначения (в одноместных командах также регистр источника);

Rr — регистр источника;

R — результат после выполнения инструкции;

K — константные данные;

k — константный адрес;

b — бит регистра общего назначения;

s — бит регистра слова состояния;

X, Y, Z — регистры косвенной адресации (указатели адреса);

A — адрес внешней памяти;

q — смещение при прямой адресации;

STACK — вершина стека;

SP — указатель вершины стека.

Систему команд можно условно разбить на группы операций: арифметическо-логические, битовые, команды передачи управления, команды пересылок. Каждая группа команд имеет определённые особенности и будет рассмотрена отдельно.

### Команды передачи управления

Большинство команд передачи управления используют относительную адресацию в памяти программ. В команды с относительной адресацией в памяти программ можно закодировать переход в пределах −63…64 байта. Следует помнить, что размеры команд могут различаться в зависимости от типа команды и её операндов, поэтому нельзя точно указать, сколько команд охватывает диапазон относительной адресации в памяти программ. Для реализации переходов, слишком «далёких» для команд с относительной адресацией, необходимо использовать косвенную адресацию при помощи указателей адреса.

Команды передачи управления, использующие относительную адресацию, содержат в начале своей мнемоники букву R (от англ. relative), например RJMP или BRNE. Команды, использующие косвенную адресацию по указателю адреса содержать букву I (от англ. indirect), например IJMP.

Всю группу команд передачи управления можно разделить на 3 подгруппы:

* команды безусловного перехода;
* команды пропуска следующей команды по условию;
* команды условного перехода.

Команды безусловного перехода включают также команды вызова подпрограммы и возврата из неё. Переход и вызов подпрограммы могут быть выполнены как с относительной, так и с косвенной адресацией. Для этих команд диапазон относительной адресации составляет слов. Таким образом, в микроконтроллере ATmega8515 косвенная адресация позволяет адресовать всю доступную память программ, а в ATmega16 — половину.

Мнемоники команд пропуска следующей команды, за исключением CPSE, начинаются с буквы S (от англ. skip) и в случае истинности условия выполняют пропуск одной следующей команды вне зависимости от её размера. Условием может являться равенство регистров общего назначения или значение любого бита регистра общего назначения или порта ввода/вывода.

Мнемоники команд условного перехода начинаются с буквы B (от англ. branch) и проверяют флаги при выполнении, а значит выполняется проверка результата выполнения предыдущей команды, модифицировавшей эти флаги. Все команды условного перехода используют относительную адресацию на −64≤*k*<64 слов. В случаях, когда требуется дальний переход, можно выполнить условный переход на команду безусловного перехода, обеспечивающую необходимую длину перехода.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Условие** | **Флаги** | **Мнемоника** | **Обратное** | **Флаги** | **Мнемоника** | **Знак** |
| Rd>Rr | *Z*&(*N*⊕*V*)=0 | BRLT | Rd≤Rr | *Z*|(*N*⊕*V*)=1 | BRGE | Знаковое |
| Rd>Rr | *C*|*Z*=0 | BRLO | Rd≤Rr | *C*|*Z*=1 | BRSH | Беззна-ковое |
| Rd≥Rr | (*N*⊕*V*)=0 | BRGE | Rd<Rr | (*N*⊕*V*)=1 | BRLT | Знаковое |
| Rd≥Rr | *C*=0 | BRSH/BRCC | Rd<Rr | *C*=1 | BRLO/BRCS | Беззна-ковое |
| Rd=Rr | *Z*=1 | BREQ | Rd≠Rr | *Z*=0 | BRNE | Любой |
| Пере-нос | *C*=1 | BRCS | Нет переноса | *C*=0 | BRCC | Любой |
| Переполне-ние | *V*=1 | BRVS | Нет переполнения | *V*=0 | BRVC | Любой |
| Отрицательное | *N*=1 | BRMI | Положи-тельное | *N*=0 | BRPL | Любой |
| Нуль | *Z*=1 | BREQ | Не нуль | *Z*=0 | BRNZ | Любой |

### Команды работы с битами

Хотя к данной группе можно также отнести и побитовые операции И, ИЛИ, исключающее ИЛИ, разработчик микроконтроллеров включает в неё только операции установки и сброса отдельных битов и операции сдвига.

Система команд предоставляет операции логического, арифметического (с учетом знакового разряда) и циклического сдвига. Также отдельные команды, позволяющие установить биты регистра слова состояния. Мнемоники команд установки бита имеют вид SEb, где b — название флага (одно из Z, N, C, V, H, I, S, T), а команд сброса бита — CLb.

### Команды пересылок

Команды пересылок содержат операции для пересылок значений между регистрами, загрузки и сохранения данных в памяти данных и памяти программ, работу с портами ввода/вывода. Команды загрузки регистра LD и сохранения значения в памяти данных ST поддерживают различные способы адресации: прямая, косвенная, косвенная с постинкрементом указателя адреса, косвенная с предекрементом указателя адреса, косвенная со смещением относительно базового адреса.

Прямая адресация позволяет получить значение ячейки памяти по непосредственно указанному адресу. При использовании данного способа адресации к мнемонике команды добавляется суффикс S. Косвенная адресация возможна по указателям адреса X, Y, Z. Адресация с пост-инкрементом также доступна по этим регистрам; её особенностью является то, что после выполнения команды используемый указатель адреса будет увеличен на 1. При адресации с пост-инкрементом в мнемонике после указателя адреса записывается символ ‘+’, например ST Y+, R3. Адресация с пре-декрементом также возможна для указателей адреса X, Y, Z; в этом случае используемый указатель перед выполнением команды будет уменьшен на 1, а в записи мнемоники необходимо добавить символ ‘-’ перед указателем адреса, например LD R5, -Z. Адресация со смешением относительного базового косвенного адреса возможна только для регистров Y и Z. При к мнемонике добавляется суффикс D, а после указателя адреса добавляется ‘+q’, где q — размер смещения, 0≤*q*≤63. Например LDD R2, Y+5. При помощи этих команд можно загружать как как регистры общего назначения, так и служебные регистры, в том числе и регистры указателей адреса. Загрузка регистра указателя адреса при использовании этого же указателя приводит к неопределённым действиям, и потому запрещена.

Описание команд микроконтроллера AVR дано ниже в таблице.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Мнемоника** | | | **Операции** | **Флаги** | **Циклы** | |
| **Арифметические команды** | | | | | | |
| ADD | | Rd, Rr | Сложение двух регистров  Rd Rd + Rr | Z C N V H | 1 | |
| ADC | | Rd, Rr | Сложение двух регистров с учетом переноса  Rd Rd + Rr + C | Z C N V H | 1 | |
| ADIW | | Rdl, K | Сложить слово и непосредственный операнд  Rdh:Rdl Rdh:Rdl + K | Z C N V S | 2 | |
| SUB | | Rd, Rr | Вычитание одного регистра из другого  Rd Rd - Rr | Z C N V H | 1 | |
| SUBI | | Rd, K | Вычитание константы из регистра  Rd Rd - K | Z C N V H | 1 | |
| SBC | | Rd, Rr | Вычитание одного регистра из другого с учётом заёма  Rd Rd - Rr - C | Z C N V H | 1 | |
| SBCI | | Rd, K | Вычитание константы из регистра с учётом заёма  Rd Rd - K - C | Z C N V H | 1 | |
| SBIW | | Rdl, K | Вычитание из слова непосредственного операнда  Rdh:Rdl Rdh:Rdl - K | Z C N V S | 2 | |
| AND | | Rd, Rr | Логическое И регистров  Rd Rd & Rr | Z N V | 1 | |
| ANDI | | Rd, K | Логическое И регистра и константы  Rd Rd & K | Z N V | 1 | |
| OR | | Rd, Rr | Логическое ИЛИ регистров  Rd Rd — Rr | Z N V | 1 | |
| ORI | | Rd, K | Логическое ИЛИ регистра и константы  Rd Rd — K | Z N V | 1 | |
| EOR | | Rd, Rr | Логическое исключающее ИЛИ регистров (сумма по модулю 2)  Rd Rd  Rr | Z N V | 1 | |
| COM | | Rd | Инверсия всех битов (преобразование в обратный код)  Rd $FF - Rd | Z C N V | 1 | |
| NEG | | Rd | Преобразование в дополнительный код  Rd $00 - Rd | Z C N V H | 1 | |
| SBR | | Rd,K | Установка битов в регистре по маске K  Rd Rd — K | Z N V | 1 | |
| CBR | | Rd,K | Сброс битов в регистре по маске К  Rd Rd & ($FF - K) | Z N V | 1 | |
| INC | | Rd | Инкремент регистра  Rd Rd + 1 | Z N V | 1 | |
| DEC | | Rd | Декремент регистра  Rd Rd - 1 | Z N V | 1 | |
| TST | | Rd | Проверка на равенство нулю регистра  Rd Rd & Rd | Z N V | 1 | |
| CLR | | Rd | Сброс всех битов регистра в 0  Rd $0 | Z N V | 1 | |
| SER | | Rd | Установка всех битов регистра в 1  Rd $FF |  | 1 | |
| MUL | | Rd, Rr | Умножение беззнаковых операндов  R1:R0 Rd  Rr | Z C | 2 | |
| MULS | | Rd, Rr | Умножение операндов со знаком  R1:R0 Rd  Rr | Z C | 2 | |
| MULSU | | Rd, Rr | Умножение числа со знаком (первый операнд) на беззнаковое число (второй операнд)  R1:R0 Rd  Rr | Z C | 2 | |
| FMUL | | Rd, Rr | Беззнаковое умножение дробных частей чисел  R1:R0 (Rd  Rr) « 1 | Z C | 2 | |
| FMULS | | Rd, Rr | Знаковое умножение дробных частей чисел  R1:R0 (Rd  Rr) « 1 | Z C | 2 | |
| FMULSU | | Rd, Rr | Умножение знаковой дробной части одного числа на беззнаковую дробную часть другого  R1:R0 (Rd  Rr) « 1 | Z C | 2 | |
| **Команды ветвления** | | | | | | |
| RJMP | | k | Относительный переход на адрес (метку)  PC PC + k + 1 |  | 2 | |
| IJMP | |  | Косвенный переход на (Z)  PC Z |  | 2 | |
| RCALL | | k | Относительный вызов подпрограммы  PC PC + k + 1 |  | 3 | |
| ICALL | |  | Косвенный вызов подпрограммы в (Z)  PC Z |  | 3 | |
| RET | |  | Возврат из подпрограммы  PC STACK |  | 4 | |
| RETI | |  | Возврат из обработчика прерывания  PC STACK | I | 4 | |
| CPSE | | Rd,Rr | Пропустить следующую команду, если регистры равны  if (Rd = Rr) PC PC + 2 (или 3)  else PC PC + 1 else PC PC + 1 |  | 1/2/3 | |
| CP | | Rd,Rr | Сравнение регистров  Rd - Rr | Z N V C H | 1 | |
| CPC | | Rd,Rr | Сравнение регистров с учетом переноса  Rd - Rr - C | Z N V C H | 1 | |
| CPI | | Rd,K | Сравнение регистра с константой  Rd - K | Z N V C H | 1 | |
| SBRC | | Rr, b | Пропустить следующую команду, если указанный бит регистра сброшен  if (Rr(b)=0) PC PC + 2 (или 3)  else PC PC + 1 |  | 1/2/3 | |
| SBRS | | Rr, b | Пропустить следующую команду, если указанный бит регистра установлен  if (Rr(b)=1) PC PC + 2 (или 3)  else PC PC + 1 |  | 1/2/3 | |
| SBIC | | P, b | Пропустить следующую команду, если указанный бит регистра ввода/вывода сброшен  if (P(b)=0) PC PC + 2 (или 3)  else PC PC + 1 |  | 1/2/3 | |
| SBIS | | P, b | Пропустить следующую команду, если указанный бит регистра ввода/вывода установлен  if (P(b)=1) PC PC + 2 (или 3)  else PC PC + 1 |  | 1/2/3 | |
| BRBS | | s, k | Относительный переход, если установлен s-тый бит регистра слова состояния  if (SREG(s) = 1) PCPC+k + 1  else PC PC + 1 |  | 1/2 | |
| BRBC | | s, k | Относительный переход, если сброшен s-тый бит регистра слова состояния  if (SREG(s) = 0) PCPC+k + 1  else PC PC + 1 |  | 1/2 | |
| BREQ | | k | Относительный переход, если операнды предыдущей операции равны или установлен флаг Z  if (Z = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRNE | | k | Относительный переход, если операнды предыдущей операции не равны или флаг Z сброшен  if (Z = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRCS | | k | Относительный переход если был перенос  if (C = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRCC | | k | Относительный переход, если не было переноса  if (C = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRSH | | k | Относительный переход, если в последней операции сравнения или вычитания первый беззнаковый операнд больше или равен второму  if (C = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRLO | | k | Относительный переход, если в последней операции сравнения или вычитания первый беззнаковый операнд меньше второго  if (C = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRMI | | k | Относительный переход, если результат последней операции отрицателен  if (N = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRPL | | k | Относительный переход, если результат последней операции положителен  if (N = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRGE | | k | Относительный переход, если в последней операции сравнения или вычитания первый операнд больше или равен второму с учетом знака  if (N  V= 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRLT | | k | Относительный переход, если в последней операции сравнения или вычитания первый операнд меньше второго с учетом знака  if (N  V= 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRHS | | k | Относительный переход, если был перенос между тетрадами  if (H = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRHC | | k | Относительный переход, если не было переноса между тетрадами  if (H = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRTS | | k | Относительный переход, если установлен флаг Т  if (T = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRTC | | k | Относительный переход, если флаг Т сброшен  if (T = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRVS | | k | Относительный переход, если в последней операции имело место переполнение  if (V = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRVC | | k | Относительный переход, если в последней операции переполнения не было  if (V = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRIE | | k | Относительный переход, если прерывания разрешены  if ( I = 1) PC PC + k + 1  else PC PC + 1 |  | 1/2 | |
| BRID | k | | Относительный переход, если прерывания запрещены  if ( I = 0) PC PC + k + 1  else PC PC + 1 |  | 1/2 |
| **Команды пересылок** | | | | | |
| MOV | Rd, Rr | | Пересылка между регистрами  Rd Rr |  | 1 |
| MOVW | Rd, Rr | | Пересылка слова из одной пары регистров в другую  Rd+1:Rd Rr+1:Rr |  | 1 |
| LDI | Rd, K | | Загрузка константы в регистр  Rd K |  | 1 |
| LD | Rd, X | | Загрузка в регистр косвенно адресуемого указателем (Х) значения  Rd (X) |  | 2 |
| LD | Rd, X+ | | Загрузка в регистр косвенно адресуемого (X) значения и пост-инкремент указателя Х  Rd (X), X X + 1 |  | 2 |
| LD | Rd, - X | | Декремент указателя Х и загрузка в регистр косвенно адресуемого (Х) значения  X X - 1, Rd (X) |  | 2 |
| LD | Rd, Y | | Загрузка в регистр косвенно адресуемого указателем (Y) значения  Rd (Y) |  | 2 |
| LD | Rd, Y+ | | Загрузка в регистр косвенно адресуемого (Y) значения и пост-инкремент указателя Y  Rd (Y), Y Y + 1 |  | 2 |
| LD | Rd, - Y | | Декремент указателя Y и загрузка в регистр косвенно адресуемого (Y) значения  Y Y - 1, Rd (Y) |  | 2 |
| LDD | Rd,Y+q | | Загрузка в регистр значения, адресуемого смещением q от базового адреса Y  Rd (Y + q) |  | 2 |
| LD | Rd, Z | | Загрузка в регистр косвенно адресуемого указателем (Z) значения  Rd (Z) |  | 2 |
| LD | Rd, Z+ | | Загрузка в регистр косвенно адресуемого (Z) значения и пост-инкремент указателя Z  Rd (Z), Z Z+1 |  | 2 |
| LD | Rd, -Z | | Декремент указателя Z и загрузка в регистр косвенно адресуемого (Z) значения  Z Z - 1, Rd (Z) |  | 2 |
| LDD | Rd, Z+q | | Загрузка в регистр значения, адресуемого смещением q от базового адреса Z  Rd (Z + q) |  | 2 |
| LDS | Rd, k | | Загрузка в регистр значения из прямо адресуемого байта области данных  Rd (k) |  | 2 |
| ST | X, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Х), данных регистра  (X) Rr |  | 2 |
| ST | X+, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Х), данных регистра и пост-инкремент указателя Х  (X) Rr, X X + 1 |  | 2 |
| ST | - X, Rr | | Декремент указателя Х и сохранение в области данных, адресуемой косвенно указателем (Х), данных регистра  X X - 1, (X) Rr |  | 2 |
| ST | Y, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Y), данных регистра  (Y) Rr |  | 2 |
| ST | Y+, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Y), данных регистра и пост-инкремент указателя Y  (Y) Rr, Y Y + 1 |  | 2 |
| ST | - Y, Rr | | Декремент указателя Y и сохранение в области данных, адресуемой косвенно указателем (Y), данных регистра  Y Y - 1, (Y) Rr |  | 2 |
| STD | Y+q,Rr | | Сохранение в области данных по базовому адресу Y со смещением q данных регистра  (Y + q) Rr |  | 2 |
| ST | Z, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Z), данных регистра  (Z) Rr |  | 2 |
| ST | Z+, Rr | | Сохранение в области данных, адресуемой косвенно указателем (Z), данных регистра и пост-инкремент указателя Z  (Z) Rr, Z Z + 1 |  | 2 |
| ST | -Z, Rr | | Декремент указателя Z и сохранение в области данных, адресуемой косвенно указателем (Z), данных регистра  Z Z - 1, (Z) Rr |  | 2 |
| STD | Z+q,Rr | | Сохранение в области данных по базовому адресу Z со смещением q данных регистра  (Z + q) Rr |  | 2 |
| STS | k, Rr | | Сохранение данных регистра Rr по прямому адресу области данных  (k) Rr |  | 2 |
| LPM |  | | Загрузка регистра R0 по косвенному адресу (Z) из памяти программ  R0 (Z) |  | 3 |
| LPM | Rd, Z | | Загрузка регистра Rd значением из памяти программ по косвенному адресу (Z)  Rd (Z) |  | 3 |
| LPM | Rd, Z+ | | Загрузка регистра Rd значением из памяти программ по косвенному адресу (Z) и пост-инкремент Z  Rd (Z), Z Z+1 |  | 3 |
| SPM |  | | Сохранение слова в памяти программ по косвенному адресу (Z)  (Z) R1:R0 |  | - |
| IN | Rd, P | | Ввод в регистр Rd из порта P  Rd P |  | 1 |
| OUT | P, Rr | | Вывод в порт P из регистра Rr  P Rr |  | 1 |
| PUSH | Rr | | Запись регистра Rr в стек  STACK Rr |  | 2 |
| POP | Rd | | Выталкивание значения из стека в регистр Rd  Rd STACK |  | 2 |
| **Команды работы с битами** | | | | | |
| SBI | P,b | | Установить бит порта ввода/вывода  I/O(P,b) 1 |  | 2 |
| CBI | P,b | | Сбросить бит порта ввода/вывода  I/O(P,b) 0 |  | 2 |
| LSL | Rd | | Логический сдвиг регистра Rd влево  Rd(n+1) Rd(n), Rd(0) 0 | Z C N V | 1 |
| LSR | Rd | | Логический сдвиг регистра Rd вправо  Rd(n) Rd(n+1), Rd(7) 0 | Z C N V | 1 |
| ROL | Rd | | Циклический сдвиг регистра Rd влево через флаг переноса  Rd(0)C,Rd(n+1) Rd(n),CRd(7) | Z C N V | 1 |
| ROR | Rd | | Циклический сдвиг регистра Rd вправо через флаг переноса  Rd(7)C,Rd(n) Rd(n+1),CRd(0) | Z C N V | 1 |
| ASR | Rd | | Арифметический сдвиг регистра Rd вправо  Rd(n) Rd(n+1), n=0..6 | Z C N V | 1 |
| SWAP | Rd | | Обмен тетрад регистра Rd  Rd(3..0)Rd(7..4),Rd(7..4)Rd(3..0) |  | 1 |
| BSET | s | | Установка s-го бита регистра слова состояния  SREG(s) 1 | SREG(s) | 1 |
| BCLR | s | | Сброс s-го бита регистра слова состояния  SREG(s) 0 | SREG(s) | 1 |
| BST | Rr, b | | Сохранение b-го бита регистра Rr во флаге T  T Rr(b) | T | 1 |
| BLD | Rd, b | | Копирование в b-тый бит регистра Rd из флага Т  Rd(b) T |  | 1 |
| SEC |  | | Установка флага переноса  C 1 | C | 1 |
| CLC |  | | Сброс флага переноса  C 0 | C | 1 |
| SEN |  | | Установка флага отрицательного результата  N 1 | N | 1 |
| CLN |  | | Сброс флага отрицательного результата  N 0 | N | 1 |
| SEZ |  | | Установка флага нуля  Z 1 | Z | 1 |
| CLZ |  | | Сброс флага нуля  Z 0 | Z | 1 |
| SEI |  | | Разрешение прерываний  I 1 | I | 1 |
| CLI |  | | Запрет прерываний  I 0 | I | 1 |
| SES |  | | Установка флага «результат со знаком»  S 1 | S | 1 |
| CLS |  | | Сброс флага «результат со знаком»  S 0 | S | 1 |
| SEV |  | | Установка флага переполнения  V 1 | V | 1 |
| CLV |  | | Сброс флага переполнения  V 0 | V | 1 |
| SET |  | | Установка флага Т  T 1 | T | 1 |
| CLT |  | | Сброс флага Т  T 0 | T | 1 |
| SEH |  | | Установка флага переноса между тетрадами  H 1 | H | 1 |
| CLH |  | | Сброс флага переноса между тетрадами  H 0 | H | 1 |

# Учебно-отладочный стенд EV8031/AVR

## Технические характеристики.

* Используемые однокристальные процессоры AT89C51, AT89C52, AT90S8515 (ATmega8515) (корпус DIP-40);
* Память программ – 16 КБайт;
* Память данных – 16 КБайт;
* Последовательная EEPROM память, 256 байт (AT24C02) в стандартной поставке;
* Два последовательных канала передачи данных RS232;
* Системный интерфейс (см. приложение №2);
* Интерфейс расширения (16 линий выход, 8 линий вход/выход, порт P1 ОЭВМ), смотрите приложение №2;
* Клавиатура 4х3
* Статическая 4-разрядная семисегментная светодиодная индикация;
* Цифроаналоговый и аналого-цифровой преобразователь (плата расширения);
* Генератор с фиксированной частотой генерации – около 10 Кгц, генератор с изменяемой частотой генерации от 1 Кгц до 50 Кгц (плата расширения);
* Динамическая 4-х разрядная семисегментная индикация (плата расширения);
* Устройство дискретного ввода информации: 2 кнопки;
* Статическая светодиодная индикация, 8 шт.;
* Знакосинтезирующий светодиодный индикатор 5x7 1 шт. (плата расширения);

## Описание учебно-отладочного стенда

### Работа стенда при загрузке отлаживаемой программы.

Схема электрическая принципиальная учебно-отладочного стенда прилагается. Перечень интегральных микросхем, а также их аналоги используемые в стенде - в таблице №1

Программа загрузчик находится в Flash-памяти микроконтроллера AT89C51, она проводит инициализацию последовательного приемопередатчика ОЭВМ (DD1), проверяет наличие и емкость памяти данных.

Память ОЗУ объемом 32К делится на две части по 16К. Одна часть для памяти программ, другая для памяти данных. В режиме загрузки вся память 32К отображается в адресное пространство, как память данных.

При поступлении данных с последовательного порта персонального компьютера в последовательный порт (разъем Х11) стенда, ОЭВМ записывает их в ОЗУ отведенное под память программ. Сигналы управления - PME, WR, RD, ALE, формируемые процессором и необходимые для обращения к памяти данных поступают через системный контроллер. После принятия последнего байта загрузчик формирует сигнал запуска программы, записью управляющего кода в системный контроллер.

Кнопка SW2, необходима для формирования сигнала сброса на входе RESET процессора, т.е. перевода стенда в режим загрузки и ожидания приема данных с последовательного порта. Процессор готов принимать данные в память данных.

Кнопка SW1, необходима для перезапуска загруженной с ПК программы находящейся в памяти программ (DD3). При нажатии кнопки SW1, загорается светодиод HL9. При этом возможна новая запись программы в стенд с персонального компьютера. При передаче данных с персонального компьютера в стенд, компьютер на линии RI последовательного порта формирует сигнал, который через системный контроллер сбрасывает процессор, также как и кнопка SW2.

### Распределение памяти стенда.



Рисунок .

Адресация (обращение) процессора к периферийным устройствам стенда реализовано как адресация к ячейкам памяти в адресном пространстве от 8000Н до FFFFH. Сигналы выборки периферийных устройств формируются дешифратором адреса внутри микросхемы системного контроллера DD4.

### Структурная схема стенда



Рисунок . Структурная схема стенда

Вся логика стенда реализована на программируемой логической микросхеме EPM7128STC100 (DD4). Системный контроллер управляет режимами работы, выработки управляющих сигналов на ОЗУ, регистры защелки, динамическим светодиодным индикатором, клавиатурой.

### Последовательный приемопередатчик.

Модуль последовательной связи сформирован на микросхеме приемника 1489, передатчика 74HC04, мультиплексора канала передачи (внутри системного контроллера).

Скорость обмена по последовательному порту **в режиме загрузки 9600б/с**. Скорость обмена по последовательному порту в отлаживаемой программе может быть изменена.

Выбор канала последовательной передачи осуществляется сигналами CFG0, CFG1 по адресу 9001Н. Установка этих битов в "логический ноль" включает порт 1, на схеме Х11, этот порт имеет неполный набор сигналов (RxD, TxD, RI) и предназначен для записи программы в стенд.

Программная установка сигналов CFG0 в "0", а CFG1 в "1" формирует выборку дополнительного канала последовательной передачи данных, разъем Х12. Дополнительный последовательный канал имеет полный набор сигналов интерфейса RS-232C.

Таблица . Карта портов ввода/вывода стенда

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Адрес | Тип цикла | B7 | B6 | | | B5 | B4 | | B3 | | B2 | B1 | | | B0 | Имя |
| Порты периферийных устройств | | | | | | | | | | | | | | | | |
| 8xx0 | Запись | [Порт A] | | | | | | | | | | | | | | PA\_REG |
| 8xx1 | Запись | [Порт B] | | | | | | | | | | | | | | PB\_REG |
| 8xx2 | Запись | [Порт C] | | | | | | | | | | | | | | PC\_REG |
| 8xx3 | Запись | x | | x | x | | | x | | x | TRISC | | x | x | | TRIS |
| ЖКИ | | | | | | | | | | | | | | | | |
| 8xx4 | Запись | Регистр команд ЖК индикатора | | | | | | | | | | | | | | LCD\_CMD |
| 8xx5 | Запись | Регистр данных ЖК индикатора | | | | | | | | | | | | | | LCD\_DATA |
| Последовательный порт | | | | | | | | | | | | | | | | |
| 9xxx | Чтение | CTS | DSR | | | DCD | RI | | KL3 | | KL2 | KL1 | | | KL0 | US\_REG |
| Cxx0 | Запись | x | x | | | X | x | | DTR | | RTS | CFG1 | | | CFG0 | UC\_REG |
|  | | | | | | | | | | | | | | | | |
| CFG1 | CFG0 |  | | | | | | | | | | | | | | |
| 0 | 0 | RS-232 | | | | | | | СОМ1, Х11 | | | | | | | |
| 0 | 1 | RS-232 | | | | | | | СОМ2, Х12 | | | | | | | |
| 1 | 0 | RS-485 | | | | | | | Прием, Х13 | | | | | | | |
| 1 | 1 | RS-485 | | | | | | | Передача, Х13 | | | | | | | |
| Индикатор и светодиоды | | | | | | | | | | | | | | | | |
| Axx0 | Запись | [Регистр индикатора 0] | | | | | | | | | | | | | | DISPLAY[0] |
| Axx1 | Запись | [Регистр индикатора 1] | | | | | | | | | | | | | | DISPLAY[1] |
| Axx2 | Запись | <зарезервировано> | | | | | | | | | | | | | | DISPLAY[2] |
| Axx3 | Запись | <зарезервировано> | | | | | | | | | | | | | | DISPLAY[3] |
| Axx4 | Запись | DP3 | DP2 | | | DP1 | DP0 | | BL3 | | BL2 | BL1 | | | BL0 | DC\_REG |
| Axx5 | Запись | <зарезервировано> | | | | | | | | | | | | | | EDC\_REG |
| Axx6 | Запись | LED7 | LED6 | | | LED5 | LED4 | | LED3 | | LED2 | LED1 | | | LED0 | LED\_REG |
| Управление работой | | | | | | | | | | | | | | | | |
| Axx7 | Запись | x | x | | | X | x | | x | | x | x | | | RUN | SYS\_CTL |
|  |  |  |  | | |  |  | |  | |  |  | | |  |  |
| Совместимые регистры | | | | | | | | | | | | | | | | |
| Bxx0 | Запись | [Регистр индикатора 1] | | | | | | | | | | | | | | DISPLAYB |
| F000 | Запись |  |  | | |  |  | |  | |  |  | | |  | DAC | nCS7 |

Таблица . Перечень комплектующих микросхем

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Обозначение на схеме | Обозначение  (импортн.) | Аналог (ближайший) | Краткое описание ИМС |
| 1 | DD1 | AT89C52  AT89S52 | нет (КФ1830ВЕ751) | Однокристальная ЭВМ |
| 2 | DD2, DD7, DD9, DD11, DD17 | 74HC573N | 1533 ИР33 | 8-разрядный регистр |
| 3 | DD3 | 62256 | нет (К537РУ17) | Стат. ОЗУ 32Кб |
| 4 | DD4 | EPM7128STC100 | нет | Прогр. Лог. Схема |
| 5 | DD18 | 74HC04 | К1564ЛН2 | 6 КМОП инверторов |
| 6 | DD10 | 1489 | К559ИП20 | Преобразователь уровня RS-232C |
| 7 | DD12 | ADM485 | неизвестен | Преобразователь уровня RS-485 |
| 8 | DD14 | AT29C02 | нет | ЭСПЗУ 2 Кбит |
| 9 | DD15 | DS1621 | нет | Цифр. темп. Датчик |
| 10 | DD16 | DS1302 | нет | Часы реального врем. |
| 11 | DD23, DD24 | ADM202 | нет | Преобразователь уровня RS-232C |

### Светодиодный индикатор

Четырехразрядный семисегментный светодиодный индикатор подключен к системному контроллеру, который автоматически выполняет динамическую регенерацию и декодирование двоичного кода в код семисегментного индикатора. Индикатор работает всегда, сразу после подачи питания. Контроллер индикатора содержит два восьмиразрядных регистра, содержимое которых отображается на индикаторе. Содержимое регистра с адресом 0xA000 отображается на двух левых разрядах, содержимое регистра с адресом 0xA001(0xB000) – на двух правых разрядах в шестнадцатеричной форме. Управление десятичными точками и гашением осуществляется через регистр DC\_REG(0xA004). Биты DP3..DP0 управляют десятичными точками. Запись 1 в соответствующий разряд включает десятичную точку. Биты BL3..BL0 управляют гашением разрядов индикатора. Запись 1 в эти биты вызывает гашение соответствующего разряда индикатора.

### Матричная клавиатура

Состояние столбца матрицы клавиатуры считывается из ячейки с базовым адресом 0x9000, биты 3..0. Соответствующий столбец выбирается нулем в разрядах адреса A2..A0. То есть, адрес 0x9006 выбирает первый столбец, адрес 0x9005 – второй столбец, адрес 0x9003 – третий столбец. Признак нажатой кнопки считывается как ноль в соответствующем разряде.

### Включение портов ОЭВМ и EEPROM памяти

Линии P1.0, P1.1 ОЭВМ могут быть отключены от внутренней периферии стенда (Шина I2C) перемычками X14, X15. На разъем интерфейса расширения эти сигналы приходят, минуя перемычки.

### Расположение элементов, назначение разъемов и перемычек

Х1 – Системный интерфейс с полным адресным пространством;

Х10 – Интерфейс расширения для подключения внешних устройств с использованием параллельного интерфейса;

Х11 – Интерфейс последовательного порта СОМ1 для связи стенда с РС;

Х12 – Интерфейс последовательного порта СОМ2 для связи стенда с другими устройствами имеющими стандартный порт RS232C;

Х3 – Интерфейс программирования AVR;

X14, X15 – Перемычка подключения устройств шины I2C к процессору;

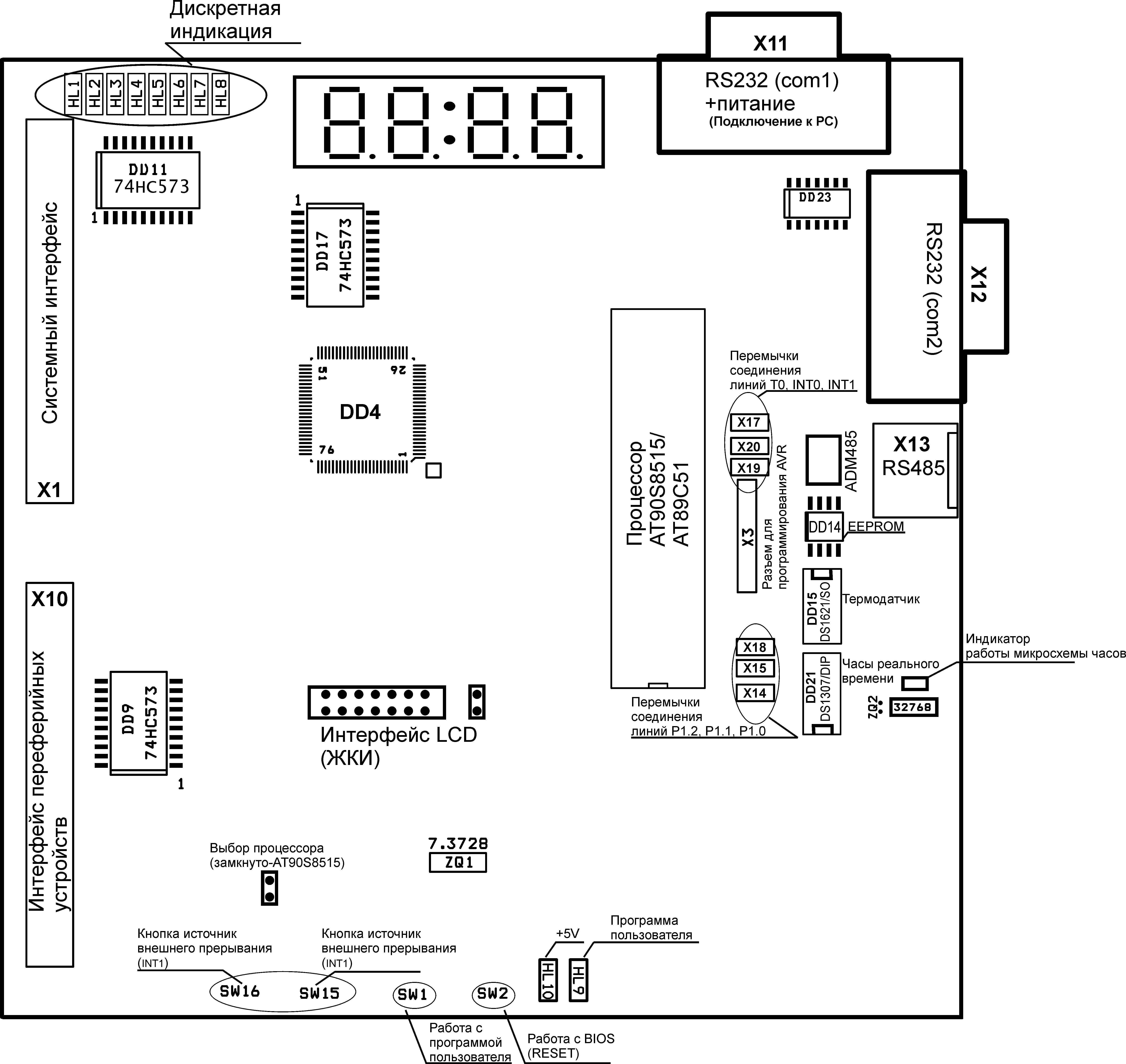


Рисунок . Схема расположения элементов стенда

## Описание плат расширения

### Назначение.

Плата расширения (в комплексе с учебно-отладочным стендом на базе однокристальной ЭВМ серии 8031) предназначена для проведения лабораторных работ связанных с аналого-цифрового и частотного преобразования, а также с обработкой дискретных сигналов. Структурная схема платы расширения приведена на рис. 3. Схема электрическая принципиальная платы расширения прилагается.

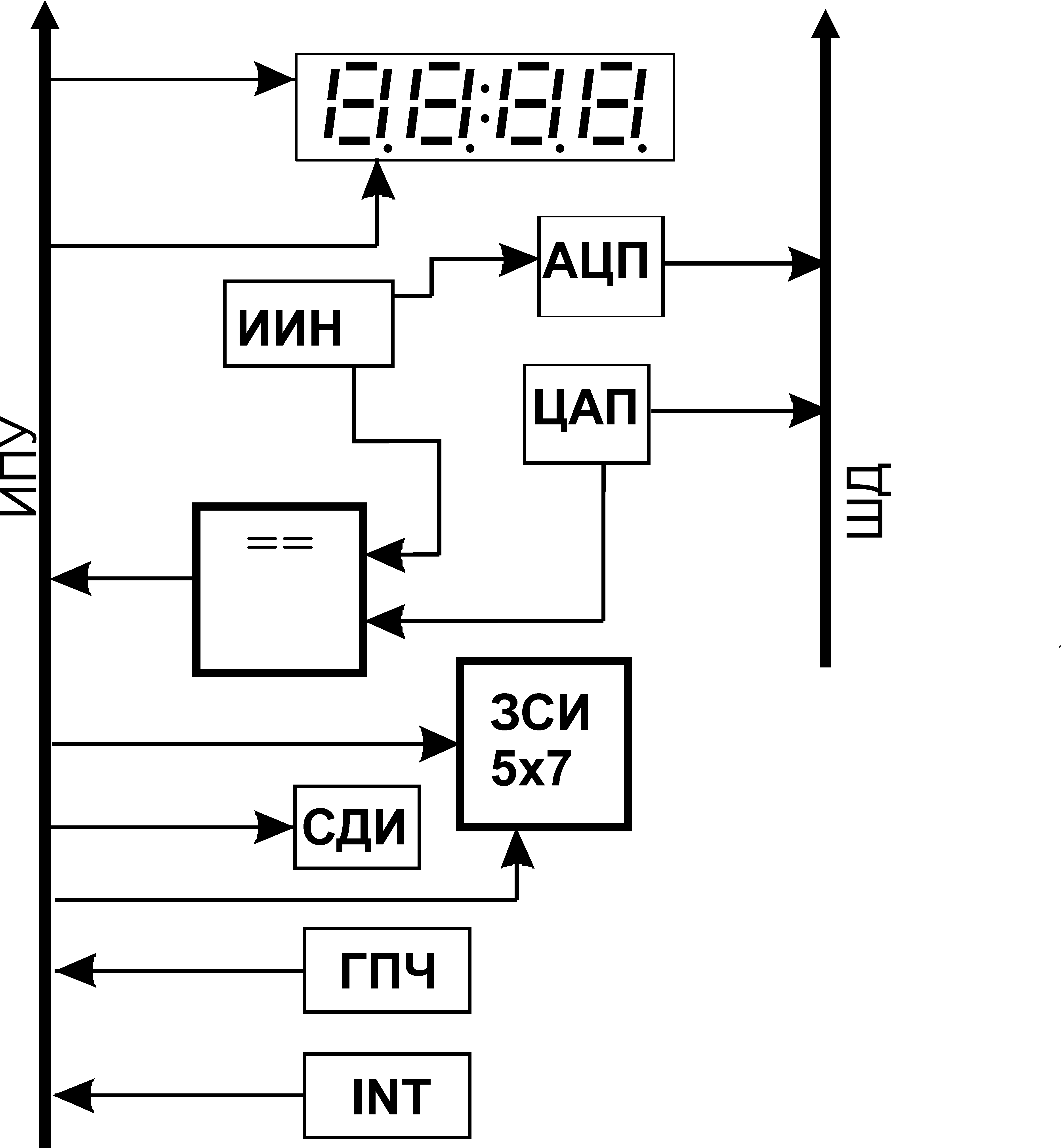


Рисунок . Структурная схема платы расширения

* 8888 – 4 разрядная динамическая индикация;
* ИПУ- Интерфейс периферийных устройств;
* ЦАП - Цифроаналоговый преобразователь;
* СДИ - Светодиодные индикаторы;
* ЗСИ – Знакосинтезирующий индикатор 5х7;
* ГПЧ – Генератор с изменяемой частотой генерации;
* INT – Кнопки запроса прерывания;
* ИИН – источник измеряемого напряжения;
* ШД – шина данных.

### Цифроаналоговый преобразователь

ЦАП выполнен на микросхеме AD7801 DD2(8 разрядный ЦАП). Входными сигналами для ЦАП являются линии AD7-AD0. Выходной сигнал снимается с разъема BNC.

### Аналого-цифровой преобразователь

АЦП выполнен на микросхеме ЦАП AD7801, операционном усилителе, используемым в качестве компаратора LM358 DA1. Входным аналоговым сигналом для АЦП являются сигнал с переменного резистора R19. Линии AD7-AD0 (см. схему стенда) используются для формирования цифрового входного кода. На выходе ЦАП формируется напряжение, пропорциональное входному коду. Сигнал срабатывания компаратора снимается с (DA1-2) вход ОЭВМ Р1.7. Срабатывание компаратора визуально видно по загоранию светодиода HL1. Если на Р1.7 "0" светодиод светится.

В расширенной комплектации стенда поставляется микросхема AD7813-8-розрядный АЦП.

### Генераторы

В схеме присутствует генератор с изменяемой частотой генерации ~ 1-50 кГц, элементы R1, R4, R5, R7, R10, R11, R15, R16, C3, VT1, DA1-1 (изменение частоты осуществляется с помощью резистора R4), и генератор с фиксированной частотой ~ 10кГц, элементы R19, R20, C16, DD18-1, DD18-2, DD18-3.

### Вывод дискретной информации

Вывод дискретной информации осуществляется с помощью четырех разрядного семисегментного индикатора HL2 включенного по схеме динамической индикации. Управление динамической индикацией осуществляется с помощью элементов DD3 (линия данных А, B, C, D, E, F, G, DP, -РВ0, РВ1, РВ2, РВ3, РВ4, РВ5, РВ6 ,РВ7) сигналы поступают с порта РВ, сигналы выборки соответствующего индикатора поступают от линий PC0, PC1 порта С.

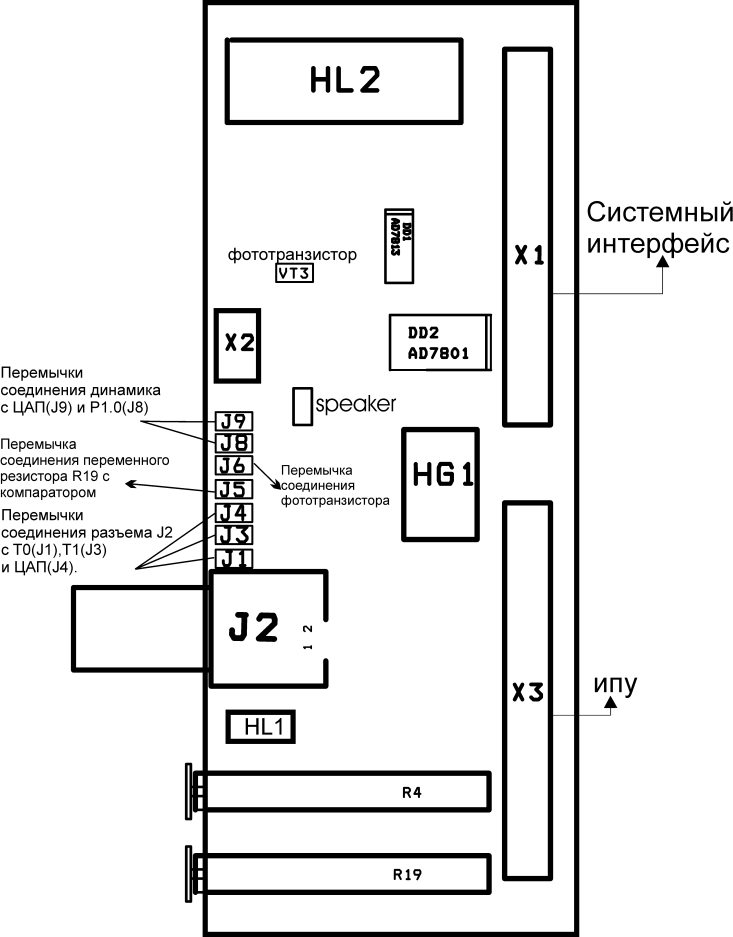


Рисунок . Схема расположения элементов платы расширения

HL2 – 4-х разрядная динамическая индикация;

HL1 – светодиодный индикатор срабатывания компаратора;

J1 – перемычка подключения к разъёму J2 выхода генератора с постоянной частотой генерации;

J2 – разъём подключения внешних контрольно-измерительных приборов.

J3 - перемычка подключения к разъёму J2 выхода генератора с изменяемой частотой генерации;

J4 – перемычка подключения к разъёму выхода ЦАП;

J5 – подключение в качестве источника внешнего прерывания INT1 кнопки S11;

J6 - подключение в качестве источника внешнего прерывания INT1 внешнего источника который может быть подключён через разъём JP1;

J7 – интерфейс подключения платы расширения к стенду;

J8 – подключение ко входу АЦП внешнего источника сигнала, подключенного к разъёму JP2;

J9 – подключение в качестве источника сигнала для АЦП переменного резистора R27;

R19 – переменный резистор, источник входного сигнала для АЦП;

R4 – переменный резистор, изменяет частоту генерации генератора импульсов;

### Плата расширения для систем автоматического управления



Рисунок . Схема расположения элементов платы расширения

JP1- Перемычка выбора вида регулирования. Выбирается регулирование частоты оборотов двигателя (MOTOR), либо регулирование температуры (LAMP).

JP2- Перемычка выбора способа регулирования. Выбирается регулирование линейное, изменяя амплитуду выходного сигнала с ЦАПа (LINE), либо регулирование ШИМ (широтно-импульсная модуляция) (PWM).

JP3- Перемычка выбора объекта регулирования. Выбирается режим регулирования частоты вращения/температуры (REG), либо режим управления звуком (SOUND).

J2, J3- Перемычки выбора способа управления звуком. 2 замкнута, 3 разомкнута - звуком управляют непосредственно в микроконтроллера. 2 разомкнута, 3 замкнута -звуком управляют при помощи ЦАПа.

X1, Х3- разъемы подключения к основной плате стенда;

U1- Датчик температуры;

U2- Датчик оборотов (На основе эффекта Холла);

LP1, LP2- нагревательные элементы (Лампы накаливания);

SW1, SW2- Дискретные кнопки;

SW3- Многооборотный переключатель;

VD1- индикатор вращения двигателя;

VD2,VD3- индикаторы нажатия кнопок SW1, SW2.

## Конструкция стенда

Учебно-отладочный стенд “EV8031”- конструктивно выполнен в виде приставки, подключаемой к персональному компьютеру посредством кабеля последовательной передачи данных и питания.

Корпусом стенду служит подложка из пластика толщиной 5мм размерами 207мм\*144мм.

## Работа со стендом

В этом разделе рассмотрен один из вариантов загрузки программы в стенд.

На персональном компьютере загрузить текстовый редактор.

В текстовом редакторе набрать текст программы в мнемокодах языка Ассемблер для 8051.

Сохранить набранный файл с расширением \*.ASM.

Откомпилировать набранную программу соответствующими средствами.

Возможные ошибки в программе можно просмотреть в одноименном файле с расширением \*.LST

После устранения всех ошибок, данные файла с расширением \*.НЕХ программой EVAL32.EXE необходимо перенести в стенд. Программа EVAL32.EXE предложит выбрать номер последовательного порта.

Вывод на экран подсказки о параметрах программы EVAL32.EXE, осуществляется запуском EVAL32.EXE.

При передаче данных с персонального компьютера в стенд на экране монитора отображаются передаваемые данные. Эти же данные отображаются на индикаторе стенда HG1. Горит светодиод HL9.

После передачи последнего байта загруженная программа запускается автоматически.

При необходимости перезапуска программы загруженной в стенд нажать кнопку SW1.

Остановка, загруженной программы и переход в режим ожидания на прием данных с персонального компьютера возможно нажатием кнопки SW2. При этом гаснет светодиод HL9.

Запись новой программы возможна в любой момент времени работы загруженной программы.

# Выполнение индивидуального задания

Согласно техническому заданию в курсовой работе необходимо создать микропроцессорную систему на основе микроконтроллера AVR (Atmega8515) и программное обеспечение для этой системы. Для тестирования микроконтроллера, а также разработанного программного обеспечения, может быть использован учебно-отладочный стенд, который описан в разделе 2 курсовой работы.

## Проектирование микропроцессорной системы

### Описание архитектуры МПС

Микропроцессорная система строится на основе микроконтроллера AVR(Atmega8515) с использованием общей шины данных и адреса. К микропроцессорной системе подключаются внешняя память данных, периферийный программируемый адаптер и внешние устройства. Централизованный контроллер прерываний внешних устройств и централизованный контроллер прямого доступа обеспечивают приоритетный переход на обработку прерываний от внешних устройств и прямой доступ внешних устройств к страницам памяти данных соответственно. Структурная схема МПС представлена на чертеже.

Рассмотрим подробнее подключаемые узлы МПС.

### Память данных (ПД)

Одной из особенностей микроконтроллеров AVR является отсутствие возможности подключения внешней памяти программ, но возможность подключения внешней памяти данных можно обеспечить. Подключение осуществляется с помощью 8-битных портов РА и РС, что позволяет уменьшить число выходов микроконтроллера. РА используется для задания младшей части адреса, а также данных для записи во внешнюю память данных. Для хранения младшей части адреса используется специальный регистр адреса, который стробируется сигналом ALE(порта РЕ1). Следует отметить что сигналы с порта РА сначала подаются на мультиплексор, так как кроме микроконтроллера с памятью и регистром адреса работает КПДП. Сигнал ALE подается на ИЛИ вмести с сигналом для записи из КПДП. Старшая часть адреса выдается непосредственно через порт РС. Запись/чтение во внешнюю память программ выполняется последовательной записью адреса в регистр и выдачей данных в следующем такте на ШАД. Выдача/приём данных стробируется сигналами записи/чтения (R/W) из микроконтроллера ИЛИ (DR/DW) из КПДП

В данной МПС к микроконтроллеру подключено 5 страниц по 64 Кб памяти каждая. Для определения микросхемы, которая в данный момент подключена к микроконтроллеру, использован дешифратор на 3 входа, который подключен к 3 младшим разрядам порта РВ. Каждая микросхема имеет 16 входов адреса (что соответствует адресному пространству в 64 Кб). Дешифратор выдает управляющие сигналы CS (chip select – выбор микросхемы), подключенные к каждой из данных микросхем. При отсутствии данного сигнала, микросхемы находятся в высокоомном состоянии и освобождают шину.

### Программируемый периферийный адаптер (ППА)

Программируемый периферийный адаптер предоставляет возможность подключить к МПС дополнительные порты, настроив их на прием или передачу данных с помощью специального управляющего слова.

ППА занимает четыре адреса во внешнем адресном пространстве памяти данных(конкретнее, он подключён к последней странице внешней памяти данных). Для реализации подключения данного устройства был использован селектор адреса (СА), настроенный на заданную четверку адресов. При совпадении адреса и сигнала CS, формируемый дешифратором номера микросхемы памяти, подается на ППА, если адрес не совпадает а сигнал СS совпадает с последней страницей внешней памяти данных – то используем память.

### Внешние устройства (ВУ)

Интерфейс внешних устройств тесно связан с реализацией механизмов прерывания и контроля прямого доступа к памяти, что будет рассмотрено дальше. В общих чертах интерфейс содержит 2 регистра, среди них: Регистр Состояния, Регистр Данных. Каждое внешнее устройство занимает 2 адреса. Первый из них закреплен за Регистрои Состояния (на чтение) и Регистром Данных (Возможно как на запись, так и на чтение).

Внешние устройства входят в адресное пространство отдельной страницы, которая выбирается с помощью дешифратора. Адреса внешних устройство от 0 до 152. Чтение и запись из/ во внешние устройства не отличается от работы с памятью.

### Режим прерываний.

Каждое внешнее устройство формирует сигнал прерывания IRQ на КПП.

КПП содержит в себе все вектора прерываний внешних устройств, регистр запросов, благодаря которому определяется какие устройства требуют прерывание, а с помощью приоритетного шифратора будет выбрано первым прерывание на обработку, которое имеет наибольшей приоритет. Также в КПП находится регистр маски, с помощью которого можна замаскировать прерывания от определённых внешних устройств(то есть даже если устройство будет запрашивать прерывание - оно обрабатываться не будет).

Если в некотором ВУ возникает запрос на прерывание, то оно выдает запрос – IRQ=1. КПП получает этот запрос и посылает к микроконтроллеру в виде сигнала INT (вход INT0). Запускается обработчик прерывания при подтверждении от микроконтроллера по сигналу INTA получаемому КПП, который разрешает ВУ выполнить прерывание выдает на ШАД вектор устройства, которое запросило прерывание, который может быть считан в обработчике прерывания, определяющий дальнейшие действия. По завершению этих действий обработчик снимает сигнал INTA и обработка прерывания считается законченной.

### Прямой доступ к памяти (КПДП)

Подобным образом работает обработка запроса прямого доступа. Соответствующие сигналы HOLD в КПДП и HOLDA в микроконтроллере выполняют запрос и подтверждение прерывания. С помощью сигналов DRQ ВУ обращаются с запросов прямого доступа к памяти к КПДП. С помощью сигналов DACK КПДП разрешает этот доступ ВУ. DR/DW служат сигналами записи чтения памяти, WE сигнал для подключения КПДП к регистру адреса. ALEDMA сигнал записи в регистр адреса. AD выставляет адрес на ШАД. Обработчик прерывания, закрепленный за входом (INT1) микроконтроллера, выполняет отключение сигналов с порта А с помощью мультикплексора. Микроконтроллер продолжает работу, запретив на программном уровне работу с внешней памятью. При окончании работы ВУ с памятью микроконтроллер снимает запрет на работу с внешней памятью данных и подключает порт РА для работы со внешними страницами памяти данных.

## Разработка микропрограмного обеспечения

По варианту задания нужно разработать программу вычисления арифметической функции. Для реализации этой функции нужно реализовать набор процедур: сложение/вычитание и умножения чисел с плавающей запятой. Процедуры реализованны на языке C. Микроконтроллер AVR не поддерживает работу с числами с плавающей запятой на аппаратном уровне. Реализация такой поддержки на програмном уровне и является главной целью.

Работа производится с числами с плавающей запятой записанных в формате оговоренном в стандарте IEEE 754. То есть на вход программы числа подаются в запакованном виде. Формат операндов: binary32 (float), то есть формат в котором на мантиссу выделено 23+1 разряд (один бит – знаковый) и на порядок 8 разрядов. По стандарту числа представлены в нормализованном виде, если это возможно.

### Разработка вспомогательных процедур

На вход программы числа подаются в запакованном виде. В таком виде работать с ними крайне не удобно, поэтому производится операция распаковки – разнесения по разным переменным порядка и мантиссы.

Формат числа записанного в формате описанном в стандарте IEEE 754 имеет следующий вид:

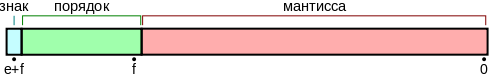


Рисунок . Формат числа с плавающей запятой

Где:

f – размер мантиссы. В нашем варианте равно 23. Следует учесть, что число подается в нормализованном виде, поэтому старший бит числа, это всегда 1. Исходя из этих соображений старший бит не хранят

e – размер порядка. В нашем варинате равно 8

Для удобства, введем собственный тип *float32*, который будет давать удобный побайтный доступ к 32 битному числу:

struct ieee\_binary32

{

uint8\_t bytes[4];

};

typedef struct ieee\_binary32 float32;

#### Распаковка

Разработано два типа распаковки:

* Распаковка в прямой код

/\*

\* Распаковка 32 битного (float) формата в прямой код (знаковый бит выделен)

\*/

void float\_unpack\_true(float32\* f, uint8\_t\* e, int32\_t\* m, uint8\_t\* s) {

uint8\_t \_b0 = f->bytes[0];

uint8\_t \_b1 = f->bytes[1];

uint8\_t \_b2 = f->bytes[2];

uint8\_t \_b3 = f->bytes[3];

\*s = \_b0 & 0x80;

\*e = ((\_b0 << 1) | (\_b1 >> 7));

if(\*e != 0)

{

\*m = 0x80 | \_b1;

}

else

{

\*m = \_b1 & 0x7f;

}

\*m <<= 8;

\*m |= \_b2;

\*m <<= 8;

\*m |= \_b3;

}

* Распаковка в прямой код (основана на распаковке в прямой код)

/\*

\* Распаковка 32 битного (float) формата в дополнительный код

\*/

void float\_unpack\_comp(float32\* f, uint8\_t\* e, int32\_t\* m) {

uint8\_t \_s;

float\_unpack\_true(f, e, m, &\_s);

if(\_s)

{

\*m = -\*m;

}

}

#### Запаковка

На выходе программы так же должно быть число представленное в формате оговоренном в стандарте. Поэтому разработанны процедуры для запаковки чисел. Их, так же как и процедур распаковки, две. В одной знак задается явно, в другой его опускают, считая что число поданно в дополнительном коде:

* запаковка числа в прямом коде

/\*

\* Запаковка числа в прямом коде в 32 битный (float) формат.

\*/

void float\_pack\_true(float32\* f, uint8\_t\* e, int32\_t\* m, uint8\_t\* s) {

int32\_t \_\_m = \*m;

uint8\_t \_\_e = \*e;

f->bytes[3] = \_\_m & 0xff;

\_\_m >>= 8;

f->bytes[2] = \_\_m & 0xff;

\_\_m >>= 8;

f->bytes[1] = (\_\_e << 7) | (\_\_m & 0x7f);

f->bytes[0] = \*s | (\_\_e >> 1);

}

* запаковка числа в дополнительном коде

/\*

\* Запаковка числа в дополнительном коде в 32 битный (float) формат.

\*/

void float\_pack\_comp(float32\* f, uint8\_t\* e, int32\_t\* m) {

do

{

if(\*m >= 0)

{

uint8\_t temp = 0;

float\_pack\_true(f, e, m, &temp);

}

else

{

uint8\_t temp = 0x80;

int32\_t tempm = -\*m;

float\_pack\_true(f, e, &tempm, &temp);

}

} while(0);

}

### Алгоритм сложения/вычитания чисел с плавающей запятой

Основной пробемой при реализации суммы числе с плавающей запятой является приведение операндова к одному порядку, а так же нормализация полученного результата. Дополнительно, нужно рассматривать множество вариантов хода алгоритма в зависимоти от результата, таких как:

* результат не нормализован
* устранимое переполнение мантиссы
* переполнение порядка
* нулевой результат

#### Блок-схема алгоритма



Рисунок . Блок схема алгоритма сложения чисел с плавающей запятой

#### Листинг программы

/\*

\* Сложение чисел в формате с плавающей запятой.

\* a, b - операнды

\* z - результат

\* Все операнды даны в запакованном виде

\*/

void add(float32\* a, float32\* b, float32\* z)

{

/\* z = a + b \*/

uint8\_t ae, be, ze;

int32\_t am, bm, zm;

float\_unpack\_comp(a, &ae, &am);

float\_unpack\_comp(b, &be, &bm);

printf("a = %d %x\n", ae, am);

printf("b = %d %x\n", be, bm);

/\* Меняем местами в случае, если порядок первого числа

\* меньше порядка второго (ae >= be).

\* Таким образом первый операнд всегда имеет больший порядок.

\*/

if(ae < be)

{

ze = ae;

ae = be;

be = ze;

zm = am;

am = bm;

bm = zm;

}

/\*

\* Сведение операндов к одному порядку (порядку первого операнда)

\*/

while(ae != be)

{

be++;

bm >>= 1;

}

/\* Сложение \*/

zm = am + bm;

ze = be;

{

/\* Проверка на переполнение и нормализация, если таковая необходима \*/

uint8\_t sign\_bits = (zm >> 24) & 3;

if(sign\_bits == 1 || sign\_bits == 2)

{

if(ze == 0xfe)

{

/\* Переполнение порядка - его нельзя устранить. Выдается ошибка \*/

error(0x01);

}

ze++;

zm >>= 1;

}

else

{

/\* Нормализация числа \*/

int32\_t high\_frac\_bit = (zm >> 1) & 0x800000;

while(((zm >> 16) & 0x80) == high\_frac\_bit && ze > 0)

{

ze--;

zm <<= 1;

}

/\* Случай нулевого результата \*/

if(zm == 0)

{

ze = 0;

}

}

}

printf("Sum result: %d %x\n", ze, zm);

float\_pack\_comp(z, &ze, &zm);

}

### Алгоритм умножения чисел с плавающей запятой по III способу

Алгоритм умножения числе с плавающей запятой использует алгоритм умножения фисел с фиксоврованной запятой, но кроме этого нужно проводить роботу с порядком результата. Он будет равен сумме порядков операндов. Именно в этом месте может произойти неустранимое переполнение порядков. Переполнения мантиссы в данном случае произойти не может, так как на результат выделяется в два раза больше места чем на каждый из операндов. Но это же влечет за собой неизбежную потерю точности, так как конечный результат обрезается до размеров операндов.

#### Функциональная схема умножения



Рисунок . Функциональная схема алгоритма умножения III способом

При умножении III способом, множитель RG3 не сдвигается, а промежуточный регистр RG1 и множимое RG2 сдвигаются налево, причем, они соединены между собой. Решение о прибавлении множителя к результату на данном шаге, принимается с учетом выдвинутого из RG2 старшего бита. В CT хранится значение счетчика, в котором первоначально записанна длина мантиссы (для формата float это 24 бита).

#### Блок-схема алгоритма умножения



Рисунок . Алгоритм умножения чисел с плавающей запятой III способом

#### Листинг программы

/\*

\* Умножение чисел в формате с плавающей запятой.

\* a, b - операнды

\* z - результат

\* Все операнды даны в запакованном виде

\*/

void mul(float32\* a, float32\* b, float32\* z)

{

/\* z = a \* b \*/

uint8\_t ae, be, ze, as, bs, zs;

int32\_t am, bm, zm;

float\_unpack\_true(a, &ae, &am, &as);

float\_unpack\_true(b, &be, &bm, &bs);

zs = as ^ bs;

/\* Переполнение порядка в одну из сторон - его нельзя устранить \*/

if ( (zs < 0) || (zs > high\_range) ) {

error (0x01);

} else {

printf("a = %d %x\n", ae, am);

printf("b = %d %x\n", be, bm);

/\* Вычисление порядка результата \*/

ze = ae + be - removal + 1;

if ( !((am == 0) && (ae == 0)) && !((bm == 0) && (be == 0)) ) {

am = am & 0x00FFFFFF;

bm = bm & 0x00FFFFFF;

zm = bm;

int i;

/\* Основной цикл умножения \*/

for (i = 0; i < mant\_len; i++) {

am = am << 1;

zm = zm << 1;

if (zm & 0x1000000) am = am | 1;

if (am & 0x800000) {

zm = zm & 0x00FFFFFF;

zm = zm + bm;

if (zm & 0x1000000) am++;

}

}

} else {

/\* Случай нулевого результата \*/

zm = 0;

ze = 0;

}

printf("Mul result: %d %x\n", ze, am);

float\_pack\_true(z, &ze, &am, &zs);

}

}

### Алгоритм вычисления основной функции

Основная функция:

То есть для ее выполнения необходимы операции сложения/вычитания, умножения (она же обеспечит возведение в квадрат). Все эти операции были реализованны в виде процедур.

#### Блок-схема алгоримта вычисления основной функции



Рисунок . Блок-схема алгоритма вычисления основной функции

#### Листинг программы

/\*

\* Программа вычисления основной функции X:= (A + B) + (A \* B) + (A^2 - B)

\* Операнды:

\* a, b - подаются в запакованном виде

\* Результат:

\* x - результат. Так же выдается в запакованном виде.

\*/

int main()

{

float32 a, b, x, y, z;

/\* Ввод первого операнда в запакованном виде \*/

/\* a = +15.5 \*/

a.bytes[0] = 0x41;

a.bytes[1] = 0x78;

a.bytes[2] = 0x00;

a.bytes[3] = 0x00;

/\* Ввод второго операнда в запакованном виде \*/

/\* b = +50 \*/

b.bytes[0] = 0x42;

b.bytes[1] = 0x48;

b.bytes[2] = 0x00;

b.bytes[3] = 0x00;

// X := A + B

add(&a, &b, &x);

// Y:= A \* B

mul (&a, &b, &y);

// Z:= X + Y

add(&x, &y, &z);

// X:= A \* A

mul (&a, &a, &x);

// B := -B

b.bytes[0] = b.bytes[0] | 0x80;

// Y = X + B

add(&x, &b, &y);

// X:= Z + Y

add(&z, &y, &x);

return 0;

}

# Выводы

Системы построенные на основе микроконтроллеров AVR являются очень актуальными в данный момент. Контроллеры этого типа очень распространены и широко применяются. Они очень удобны в работе, так как имеют возможность многократного перепрошивания и программаторы широко доступны. Линейка контроллеров AVR позволяет подобрать наиболее оптимальный контроллер под конкртную задачу, так как при очень не существенных отличиях в системе команд, выпускаются контроллеры с самым разнообразным набором периферии, а так же характеристиками памяти программ и данных. Так же эти контроллеры имеют развитые интерфейсы для подключения внешних устройств.

Для контроллеров AVR можно программировать как на ассемблере, так и на языке С. Это сильно облегчает и ускоряет разработку програмного обеспечения для этих контроллеров. Большой набор уже написанных библиотек для этих контроллеров, так же сильно помогает в разработке. Созданное в ходе написания курсовой програмное обеспечение, хороший пример создания таких библиотек. Так, при отсутствии аппаратной поодержки вычислений с плавающей запятой, эту поддержку можно реализовать на програмном уровне.

В работе так же рассмотрен стенд, для работы с контроллерами Intel 8051 и AVR ATmega8515. Стенд значительно упрощает процедуру прошивки контроллеров, а так же предоставляет широкий спектр периферийных устройств: внешние светодиоды, многострочный жидко-кристалический экран, цифровой экран, знакосинтезирующее устройство, разнообразные датчики, цифровую клавиатуру, управляющие кнопки и прочие. Такое разнообразие внешних устройств позволяет тестировать програмное обеспечение контроллера множеством способов.

Програмное обеспечение для работы с числами с плавающей запятой разрабатывалось на языке C и может быть скомпилированно как для микроконтролера, при помощи компилятора AVR-GCC, так и для x86-процессоров при помощи компилятора GCC. Разработанные алгоритмы были отлажены и протестированны в среде разработки Eclipse.

# Список использованной литературы

1. Жабін В.І., І.А. Жуков, І.А.Клименко, Ткаченко В.В. - Прикладна теорія цифрових автоматів: Навч. посібник. – К.:Книжкове вид-во НАУ, 2007. - 364 с.
2. Жабін В.І., І.А. Жуков, І.А.Клименко, Стіренко С.Г. – Арифметичні та управляючі пристрої цифрових ЕОМ: Навчальний посібник. – К.:ВЕК +, 2008. - 176 с.
3. Трамперт В. – Измерение, управление и регулирование с помощью AVR-микроконтроллеров.: Пер. с нем. – К.: «МК-Пресс», 2006. – 208 с. ил.
4. 3. В.І. Жабін, І.А. Жуков «Мікропроцесорні системи»: Навч. Посіб. – К.: Видавництво «СПД Гуральник О.Ю.» , 2009 – 492 с.
5. Самофалов К.Г., Корнійчук В.І., Тарасенко В.П. - Электронные цифровые вычислительные машины. – К. Вища школа, 1983.