Министерство образования и науки,
молодежи и спорта Украины
Севастопольский национальный технический университет
Кафедра радиотехники и телекоммуникаций

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Лабораторный практикум по дисциплине
”Цифровые устройства, ч. II ”
для студентов
направления 6.050901 — «Радиотехника»

Севастополь
2011

УДК 681.3.068
Методические указания «Лабораторный практикум по дисциплине «Цифровые устройства ч. II»/ СевНТУ; сост. А.А. Щекатурин.— Севастополь: Изд-во СевНТУ, 2011. — 52 с.

Целью методических указаний является оказание помощи студентам в выполнении лабораторных работ по второй части дисциплине «Цифровые устройства» при изучении структуры *AVR* микроконтроллеров и получении навыков их программирования.

Методические указания утверждены на заседании кафедры радиотехники и телекоммуникаций 2 декабря 2010 года, протокол № 5.

Допущено учебно-методическим центром СевНТУ в качестве методических указаний.

Рецензент:
канд. техн. наук, доцент Зиборов С.Р.

Ответственный за выпуск:
заведующий кафедрой радиотехники и
телекоммуникаций, д-р техн. наук,
профессор Гимпилевич Ю.Б

СОДЕРЖАНИЕ

Введение 4

1. Основные теоретические сведения 5

2. Работа с отладочным стендом 10

3. Лабораторная работа № 1

Изучение средств программирования и структуры микроконтроллерной системы 11

4. Лабораторная работа № 2

Программирование статических семисегментных индикаторов 15

5. Лабораторная работа № 3

Программирование точечного индикатора 19

6. Лабораторная работа № 4

Ввод данных в микроконтроллерную систему с помощью клавиатуры 24

7. Лабораторная работа № 5

Программирование внешних прерываний 29

8. Лабораторная работа № 6

Программирование таймера/счетчика 35

Библиографический список 40

Приложение А. Список ассемблерных команд *AVR* микроконтроллеров 41

Приложение Б. Схема расположения элементов основного модуля отладочного стенда 49

Приложение В. Схема расположения элементов модуля расширения отладочного стенда 51

ВВЕДЕНИЕ

Вторая часть дисциплины «Цифровые устройства» изучается студентами направления 6.050901 — «Радиотехника» дневной формы обучения в 6-м семестре и студентами заочной формы обучения в 8-м семестре.

Методические рекомендации по выполнению каждой лабораторной работы содержат следующие разделы: цель работы, теоретическую часть, порядок выполнения работы, рекомендации по оформлению отчета.

Порядок выполнения лабораторных работ

Лабораторные работы выполняются фронтальным методом, т. е. студенческая группа, разбитая на подгруппы, выполняет одну и ту же лабораторную работу, что позволяет синхронизировать изучение материала по дисциплине, а также дает возможность преподавателю работать со всей группой одновременно.

Перед выполнением лабораторной работы студенты обязаны:

— до начала занятия самостоятельно изучить методические указания к предстоящей лабораторной работе;

— в начале занятия представить и защитить отчет по предыдущей лабораторной работе;

— ответить на контрольные вопросы, связанные с выполняемой следующей работы.

Каждый студент выполняет лабораторную работу в соответствии с индивидуальным заданием на учебном отладочном стенде *EV*8031*/AVR*. Отладочный стенд построен на базе микроконтроллера *ATMega*8515 и программируется с помощью персонального компьютера. Используемый программно–аппаратный комплекс позволяет осуществлять набор программы на языке ассемблер *AVR* микроконтроллеров, выполнять ее отладку и компилирование, записывать программу в память отладочного стенда, после чего программа начинает автоматически выполняться. Правильность работы программы может быть проконтролирована по показаниям программируемых индикаторов отладочного стенда.

Защита отчета по лабораторной работе производится каждым студентом индивидуально на очередном лабораторном занятии. В процессе защиты проверяются: правильность оформления отчета, умение студентов пояснять выполненные расчеты, знание ими методики моделирования, умение пояснить полученные зависимости и обосновать выводы по работе.

Требования к отчету по лабораторной работе

Отчет по работе оформляется в соответствии с методическими указаниями по оформлению текстовых работ на кафедре РТ [1] и должен содержать:

— титульный лист;
— цель работы;
— задание;
— блок-схему алгоритма работы программы;
— программу на ассемблере;
— результаты работы программы;
— выводы по работе.

1. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Основные сведения о микроконтроллере *ATMega*8515

Микроконтроллеры *AVR* семейства Mega являются 8-битными *RISC* микроконтроллерами.

Структурная схема микроконтроллера *ATMega*8515 изображена на рис. 1.1.

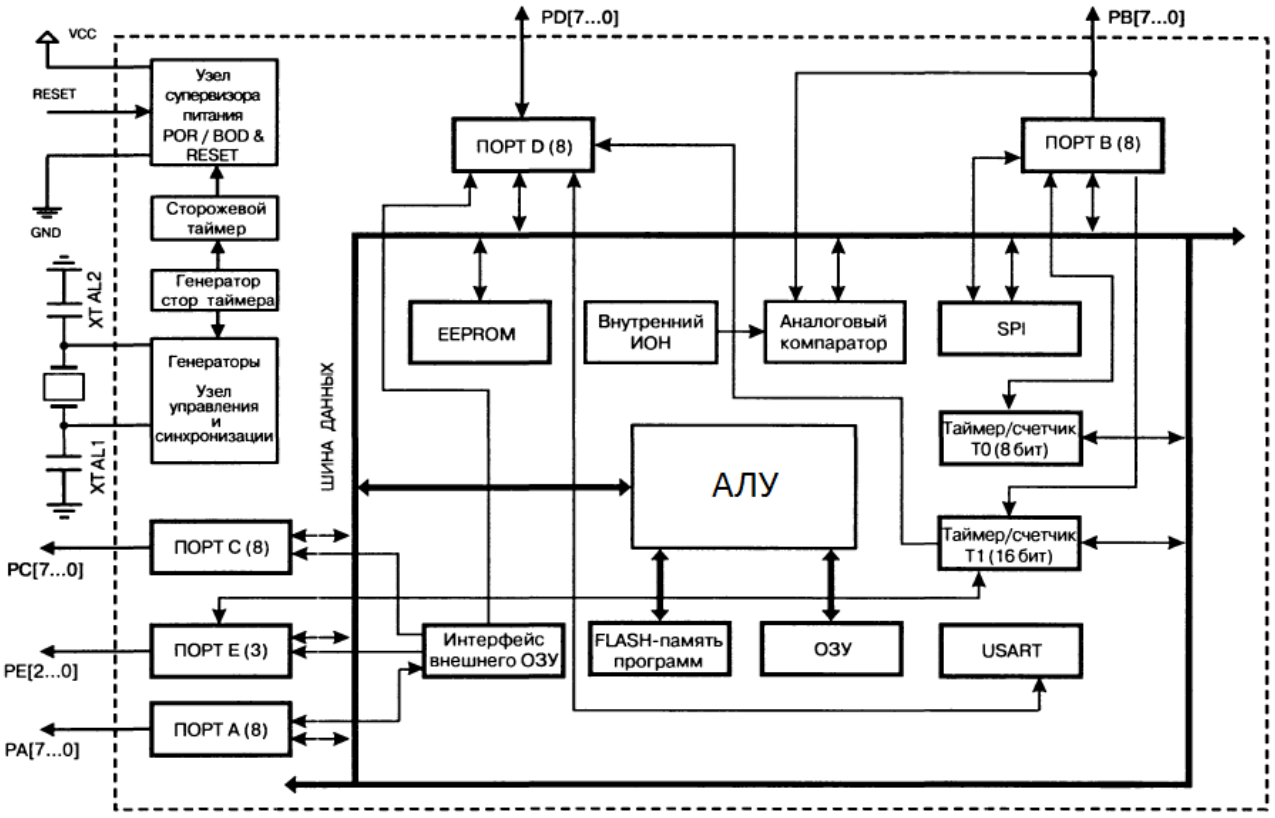


Рис. 1.1 — Структурная схема микроконтроллера *ATMega*8515

Арифметические и логические операции выполняются в арифметико-логическом устройстве (АЛУ).

Микроконтроллер содержит следующие периферийные устройства:

— 8-битный таймер/счетчик;
— 16-битный таймер/счетчик;
— сторожевой таймер;
— аналоговый компаратор;
— последовательный синхронный интерфейс *SPI*;
— полнодуплексный универсальный синхронно/асинхронный приемо-передатчик (*USART*);
— пять параллельных портов.

В аналоговом компараторе в качестве опорного может использоваться внешнее напряжение или напряжение от внутреннего источника опорного напряжения (ИОН).

Машинный цикл выполняется за один такт. Большинство команд выполняются за 1..2 машинных цикла (список команд с указанием времени выполнения команд в машинных циклах приведен в Приложении А).

В микроконтроллерах *AVR* реализована гарвардская архитектура, характеризующаяся использованием раздельной памятью программ и памятью данных

(для доступа к которым используются отдельные шины). Такая организация позволяет одновременно работать с памятью программ и памятью данных. Почти все команды (исключение составляют команды, у которых одним из операндов является 16-битный адрес) занимают одну 16-ти разрядную ячейку памяти программ.

В состав микроконтроллера входит электрически стираемая память программ (*FLASH*) и данных (*EEPROM*), а также память данных, выполненная на статическом оперативном запоминающем устройстве (ОЗУ).

Структура памяти микроконтроллера показана на рис. 1.2.

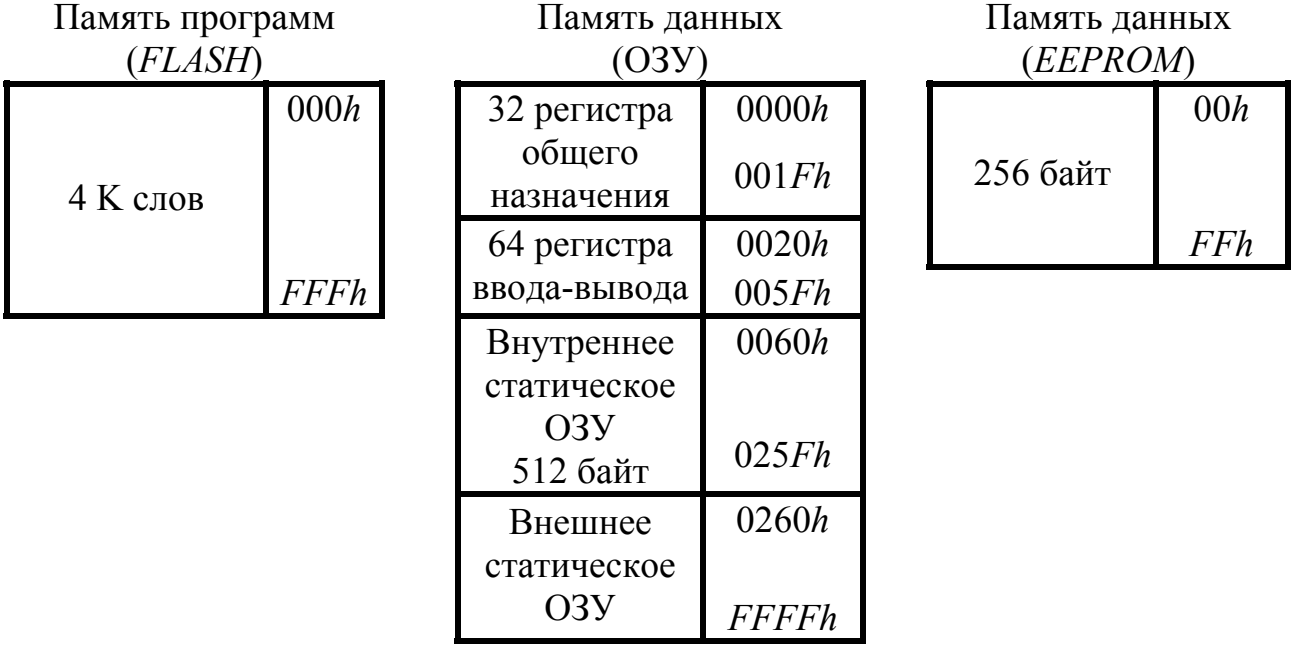


Рис. 1.2 — Структура памяти микроконтроллера *ATMega*8515

Память программ состоит из слов, каждое из которых имеет длину два байта. Память данных состоит из ячеек длиной один байт.

Особенностью микроконтроллера *ATMega*8515 является возможность использования внешнего ОЗУ.

По результатам выполнения команд в регистре состояния *SREG* (флаговый регистр) устанавливаются признаки (флаги), характеризующие то, как была выполнена команда.

Структура регистра состояний *SREG* показана на рис. 1.3.

*D*7	*D*6	*D*5	*D*4	*D*3	*D*2	*D*1	*D*0
I	*T*	*H*	*S*	*V*	*N*	*Z*	*C*

Рис. 1.3 — Структура регистра состояний *SREG*

Регистр состояний имеет следующий состав:

I — флаг прерывания; устанавливается (т. е. в разряд *D*7 записывается единица), если прерывания разрешены;

T — флаг пользователя, который устанавливается пользователем;

H — флаг полупереноса, который устанавливается, если в результате выполнения операции был перенос из младшей тетрады в старшую или произошел заем из старшей тетрады при выполнении некоторых операций;

V — флаг переполнения, который сигнализирует о выходе за допустимый диапазон результата арифметических операций над знаковыми операндами и определяется как исключающее ИЛИ между флагом переноса *C* и переносом в старший разряд *D*7;

N — флаг отрицательного значения (старший разряд числа в дополнительном коде);

S — флаг знака, состояние которого определяется результатом выплолнения операции исключающее ИЛИ между флагами *N* и *V*;

Z — флаг нулевого значения, который устанавливается, если результат выполненной операции равен нулю.

C — флаг переноса, который устанавливается, если в процессе выполнения операции произошел выход результата за пределы байта.

1.2. Основные сведения об отладочном стенде

Отладочный стенд выполнен в виде основного модуля и модуля и модуля расширения. Структурная схема основного модуля изображена на рис. 1.4.

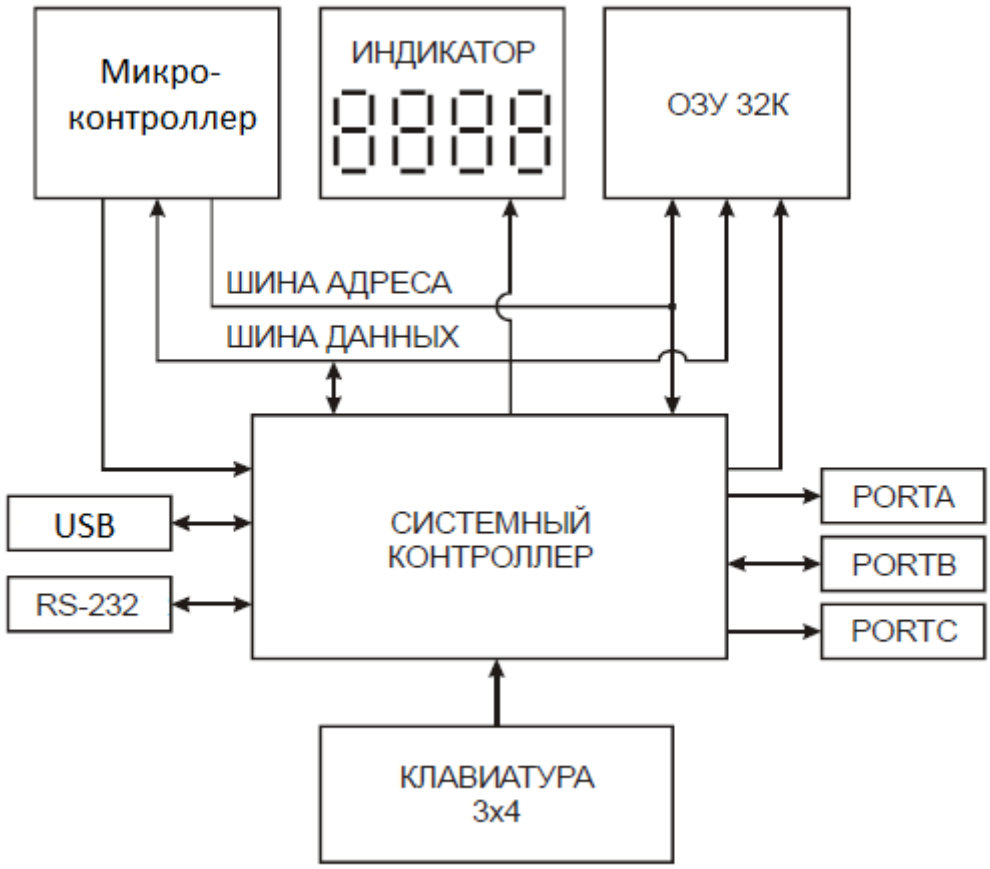


Рис. 1.4 — Структурная схема основного модуля отладочного стенда

На рис. 1.4 использованы следующие обозначения:

— Индикатор — статический четырехразрядный семисегментный индика-

тор;

— ОЗУ — оперативное запоминающее устройство;

— *PORTA*, *PORTB*, *POTRTC* — параллельные порты;

— *RS-232* — последовательный интерфейс *RS*-232;

— *USB* — *USB*-интерфейс.

Для хранения программы и данных в отладочном стенде используется внешнее статическое ОЗУ объемом 32 КБ (см. рис.1.4). Половина этой памяти используется для хранения программы, половина — для хранения данных.

Системный контроллер, выполненный на программируемой логической микросхеме *EPM*7128*STC*100 управляет режимами работы стенда, портами стенда, динамическим светодиодным индикатором и клавиатурой.

Структурная схема модуля расширения показана на рис. 1.5.

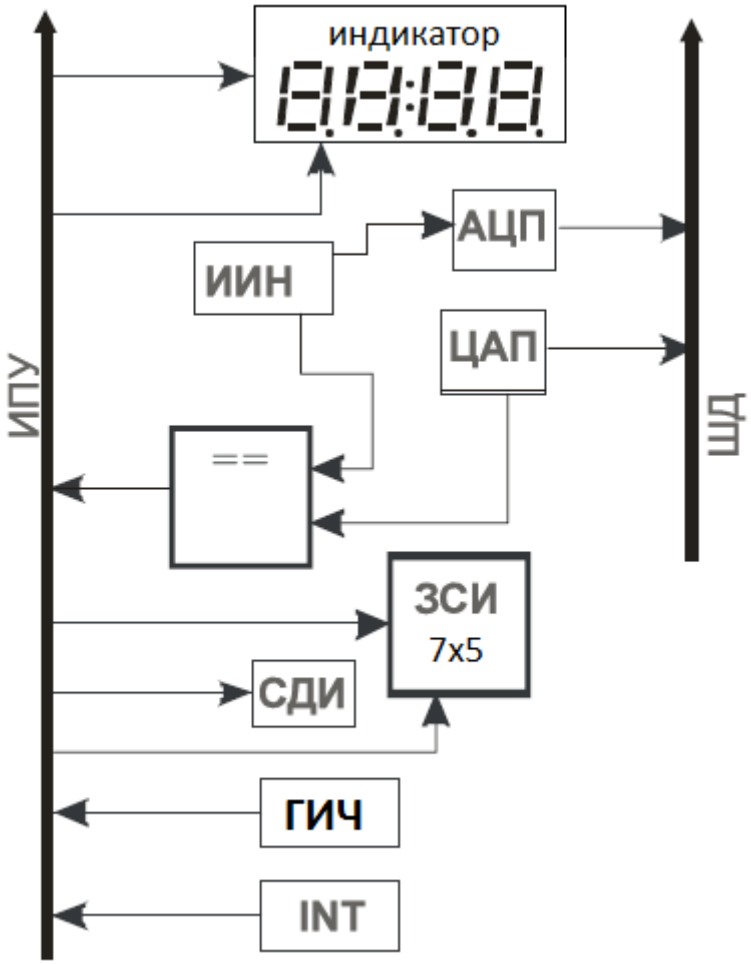


Рис. 1.5 — Структурная схема модуля расширения

На рис. 1.5 использованы следующие обозначения:

Индикатор — динамический четырехразрядный семисегментный индикатор;

ИПУ — интерфейс периферийных устройств;

АЦП — аналогово-цифровой преобразователь;

ЦАП — цифроаналоговый преобразователь;

СДИ — светодиодный индикатор;

ЗСИ — знакосинтезирующий (точечный) индикатор 7×5;

ГИЧ — генератор с изменяемой частотой генерации;

INT — кнопка запроса прерывания;

ИИН — источник изменяемого напряжения;

ШД — шина данных.

Модуль расширения содержит средства для выполнения действий, связанных с аналогово-цифровым и частотным преобразованиями, а также с обработкой дискретных сигналов.

ЦАП и АЦП восьмиразрядные. Аналоговый сигнал выводится на разъем BNC отладочного стенда.

Генератор с изменяемой частотой генерации формирует сигналы с частотой от 1 до 50 КГц.

Схема расположения элементов основного модуля и модуля расширения отладочного стенда показаны в Приложениях Б и В соответственно.

2. РАБОТА С ОТЛАДОЧНЫМ СТЕНДОМ

В соответствии с индивидуальным заданием по лабораторной работе необходимо написать программу на языке ассемблер, откомпилировать ее и ввести в память макета, после чего программа начнет выполняться автоматически.

Программа на языке ассемблер набирается в текстовом редакторе, например, в программе Блокнот *Windows*.

Набранный в текстовом редакторе файл программы необходимо сохранить в каталоге *Cu* с расширением *asm* (например, *new.asm*). Каталог *Cu*, как правило, находится в последнем разделе винчестера. Например, если винчестер разбит на разделы *c*:, *d*:, *e*:, то каталог *Cu* будет находиться в разделе *e*:. В каталоге *Cu* должны находиться файлы *avrasm.exe* и *avreal32.exe*.

Для компилирования исходного текстового программы в командной строке рекомендуется использовать *Windows commander* или *Norton Commander*. При этом необходимо следить за тем, чтобы каталог *Cu* был активен. После ввода "*avrasm.exe new.asm new.lst new.hex*" в каталоге *Cu* создается файл листинга *new.lst*, содержащий указания на ошибки в тексте программы, которые необходимо исправить. Если ошибки не обнаружены, то в каталоге *Cu* создается файл *new.hex*, который затем должен быть введен в память макета.

Если в исходном тексте программы были допущены ошибки, то нужно открыть файл листинга *new.lst*, найти отмеченные ошибки и исправить их в исходном файле *new.asm*, после чего заново его откомпилировать.

В случае, если компилирование прошло успешно, подтверждением чего является создание файла *new.hex*, который необходимо поместить в память стенда. Для этого в командную строку надо ввести "*avreal*32*.exe +MEGA*8515 *–p*1 *–e –w –v new.hex*". Здесь *+MEGA*8515 — тип программируемого микроконтроллера, *p*1 — номер *LPT* порта, *e* (*erase*) — очистить память контроллера, *w* (*write*) — записать *hex*-файл в память макета, *v* — проверить, как прошла запись. Помещенная в память отладочного стенда программа должна автоматически выполняться.

В качестве примера выполните указанные выше шаги для следующей демонстрационной программы, выводящей число 28*h* на левый статический семисегментный индикатор:

```
.DEVICE AT90s8515     ; выбираем устройство
.EQU MCUCR = 0x35     ; присваиваем регистру MCUCR адрес 0x35
.CSEG
ldi r16, 0b11000000   ; в регистр MCUCR записываем число 11000000b;
```

При этом микроконтроллер переводится в режим работы с внешней памятью, а его порты *A* и *C* используются как шины адреса и данных

```
out MCUCR, r16
;---------------------------------
ldi r16,0x28          ; записываем в регистр r16 число 28h
sts 0xA000,r16        ; выводим на левый индикатор число 28h.
```

3. ЛАБОРАТОРНАЯ РАБОТА № 1
ПРОГРАММИРОВАНИЕ СВЕТОДИОДНЫХ ИНДИКАТОРОВ

3.1. Цель работы

Целью работы является изучение способов программирования светодиодных индикаторов в микроконтроллерных системах.

3.2. Теоретические сведения

Если по адресу 0*xA*006 линейки светодиодов вывести некоторое число, то в линейке светодиодов засветятся только те светодиоды, которым будет соответствовать единица в двоичной записи этого числа. Например, если вывести число *B*5*h*, то зажгутся светодиоды **x****x***x** (* — светодиод светится, *x* — светодиод не светится), так как числу *B*5*h* в шестнадцатеричной системе счисления соответствует число 10110101*b* в двоичной системе счисления.

Шестнадцатеричные числа обозначаются с префиксом 0*x*, а двоичные с префиксом 0*b*. Например, число из предыдущего абзаца будет записано как 0*xB*5 и 0*b*10110101. Если число записано без префикса, то это означает, что это число записано в десятичной системе счисления.

Программа 3.1 — Программа, выводящая на линейку светодиодов “бегущий огонек”:

```
.DEVICE AT90s8515          ; Выбираем устройство
.EQU    MCUCR = 0x35       ; Устанавливаем для регистра управления
                           микроконтроллером MCUCR адрес 0x35
.CSEG
ldi r16,0b11000000         ; В регистр управления MCUCR заносим
                           11000000b. Тем самым переводим микроконтроллер в режим работы с внешней памятью
out MCUCR,r16              ; Теперь порты A и C используются для работы с
                           шинами адреса и данных
;---------------------------------
clr r16                ; Очищаем  порты (это не обязательно)
sts 0x8000,r16
sts 0x8001,r16
sts 0x8002,r16
sts 0xA000,r16
sts 0xA001,r16
ldi r16,1              ; В регистр r16 заносим 1 (вместо r16 можно взять любой другой свободный регистр общего назначения)
MainLoop:
sts 0xA006,r16         ; Выводится на светодиоды значение из r16
ror r16                ; Сместить вправо регистр r16 через флаг C. (Получается, что единица будет постоянно смещаться)
ldi r19,10             ; Организуем задержку с помощью трех вложенных
                       друг в друга циклов
```

```
delay3: clr r18
delay2: clr r17
delay1:
dec r17
brne delay1
dec r18
brne delay2
dec r19
brne delay3        ; Конец задержки
rjmp MainLoop      ; Переход на метку MainLoop
```

3.3. Порядок выполнения работы

3.3.1. Набрать программу 3.1 в текстовом редакторе, откомпилировать и поместить в память отладочного стенда. Убедиться, что программа выполняется правильно.

3.3.2. Внести в программу 3.1 такие изменения, чтобы “бегущий огонек” на линейке светодиодов двигался быстрее.

3.3.3. Изменить программу так, чтобы “бегущий огонек” на линейке светодиодов стал двигаться в противоположном направлении.

3.3.4. Изменить программу так, чтобы по линейке светодиодов двигались два касающихся “огонька”.

3.3.5. Внести в программу 3.1 такие изменения, чтобы “огонек” на линейке светодиодов оставался неподвижным и мигал с частотой 1 Гц.

3.3.6. В соответствии с выданным преподавателем вариантом выполнить задание из таблицы 3.1.

Таблица 3.1 — Индивидуальное задание по лабораторной работе № 1

№ п/п	Задание
1	Перемещать комбинацию **x*x*x*x* циклически слева направо со скоростью одно положение в секунду.
2	Перемещать комбинацию ***xx**xx* циклически справа налево со скоростью 2 с на одно положение.
3	Сделать так, чтобы с интервалом в 1 с сменялись две ситуации: *xx****xx* и ***xxxx***.
4	Повторять циклически с интервалом в 1 с следующие три ситуации: *xxx**xxx, xx*xx*xx, x*xxxx*x*.
5	На линейке светодиодов четыре подряд следующие “огонька” должны следовать слева направо со скоростью 1 с на позицию.
6	Правая и левая половины линейки светодиодов должны зажигаться попеременно с интервалом 2 с.
7	Сделать так, чтобы в течение 1 с светилась комбинация светодиодов **xx**x*x*, потом в течение 1 с инверсная ей, а потом все повторялось с начала.
8	Попеременно с интервалом в 1 с должны зажигаться одновременно оба

№ п/п	Задание
	крайних светодиода и одновременно два центральных светодиода.
9	Сделать так, чтобы “бегущий огонек”, занимающий одну позицию на линейке светодиодов, продвигался слева направо, каждый раз перескакивая через одну позицию. Например, сначала **xxxxxxx*, потом *xx*xxxxx*, потом *xxxx*xxx* и т. д. По достижении правого края индикатора процесс должен циклически повторяться.
10	Последовательно должны светиться три следующие комбинации *****xxxx, xx****xx*, *xxxx*****. Затем процесс должен циклически повторяться.
11	Комбинация *xx**xx*** должна циклически перемещаться слева направо со скоростью 3 с на позицию.
12	На линейке светодиодных индикаторов два крайних левых светодиода должны зажигаться попеременно с двумя крайними правыми светодиодами с интервалом в 3 с.
13	Комбинация **xx*xx*** должна оставаться неподвижной в течение 1 с, затем сдвигаться вправо, находиться в новом положении в течение 1 с, а затем вернуться в исходное положение. После этого описанное должно повторяться циклически.
14	Комбинация ***xx**xx* должна циклически сдвигаться вправо на две позиции, затем на одну позицию влево и снова вправо на две позиции и т. д.
15	Комбинация *xxxxx**** должна циклически сдвигаться влево на три позиции, затем вправо на две позиции, потом снова влево на три позиции и т. д.
16	Комбинация ***x*xxxx* должна светиться в течение 1 с, затем должна быть инвертирована и снова светиться в течение 1 с, затем должна быть сдвинута вправо на 1 позицию и светиться в течение 1 с, затем снова проинвертирована и т. д.

3.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

3.5. Вопросы для самоконтроля

3.5.1. Поясните, каким образом можно организовать циклическое выполнение программы.

3.5.2. Каким образом можно увеличить или уменьшить скорость движения огонька на линейке светодиодов?

3.5.3. Как в программе 3.1 учесть тот факт, что команды циклического сдвига осуществляют его через флаг переноса?

3.5.4. Почему линейка светодиодов имеет адрес, такой же как у ячейки внутреннего ОЗУ (а не порта, например)?

3.5.5. Какие изменения нужно внести в программу 3.1, чтобы на линейку светодиодов выводился медленно движущийся пульсирующий огонек?

3.5.6. Какие изменения нужно внести в программу 3.1, чтобы время за-

держки увеличилось в 2 раза?

3.5.7. Какие изменения нужно внести в программу 3.1, чтобы время задержки уменьшилось в 20 раз?

4. ЛАБОРАТОРНАЯ РАБОТА № 2 ПРОГРАММИРОВАНИЕ СТАТИЧЕСКИХ СЕМИСЕГМЕНТНЫХ ИНДИКАТОРОВ

4.1. Цель работы

Целью работы является изучение способов программирования статической индикации в микроконтроллерных системах.

4.2. Теоретические сведения

В отладочном стенде используются два статических семисегментных индикатора, каждый из которых способен отображать двухразрядное шестнадцатеричное число. Таким образом, в каждый индикатор может быть выведено по одному байту, левая тетрада байта будет отображена на индикаторе как левая (старшая) цифра шестнадцатеричного числа, правая тетрада — как правая (младшая) цифра шестнадцатеричного числа.

Адрес левого индикатора $A000h$, адрес правого индикатора $B000h$.

Каждый разряд индикатора может быть независимо от других зажжен или погашен. Кроме того, в каждом из четырех разрядов может быть зажжена точка (обычно используется для отделения целой части числа от дробной). Для того, чтобы зажигать и гасить цифры и точки в разрядах, используется адрес $A004h$. Единица в старшей тетраде числа, выводимого по этому адресу, зажигает точку в соответствующем разряде, а единица в младшей тетраде этого же числа гасит цифру соответствующего разряда. Например, если по адресу $A004h$ вывести число $00110101b$, то точки зажгутся в нулевом и первом (здесь разряды считаются слева направо) разрядах, а цифры в нулевом и втором разрядах будут погашены.

В программе 4.1 приведен пример программирования статических семисегментных индикаторов.

Программа 4.1

Программа обеспечивает вывод на семисегментный индикатор числа $51h$ (второй индикатор погашен), мигающего с частотой 0,5 Гц, т. е. число должно высвечиваться на индикаторе в течение 1 с, затем гаснуть на 1 с, затем процесс повторяется циклически.

```
.DEVICE AT90s8515          ; Выбираем устройство
.EQU    MCUCR = 0x35       ; Устанавливаем для регистра управления
                           микроконтроллером MCUCR адрес 0x35
.CSEG
ldi r16,0b11000000         ; В регистр управления MCUCR заносим
                           11000000. Тем самым переводим микроконтроллер в режим работы с внешней памятью
out MCUCR,r16              ; Теперь порты A и C используются для работы с
                           шинами адреса и данных
```

```
;-------------------------------------
MainLoop:
ldi r16,0x51
sts 0xA000,r16          ; Выводим на левый индикатор число 51h
ldi r16,0b00001100      ; Зажигаем оба разряда левого индикатора
sts 0xA000,r16          ; Выводим на левый индикатор число 51h
ldi r19,10      ; Организуем задержку с помощью трех вложенных друг в
                друга циклов
delay3: clr r18
delay2: clr r17
delay1:
dec r17
brne delay1
dec r18
brne delay2
dec r19
brne delay3     ; Конец задержки
ldi r16,0b00001111      ; Гасим все разряды индикатора
ldi r19,10      ; Организуем задержку с помощью трех вложенных друг в
                друга циклов
delay31: clr r18
delay21: clr r17
delay11:
dec r17
brne delay11
dec r18
brne delay21
dec r19
brne delay31        ; Конец задержки
rjmp MainLoop       ; Переходим на метку MainLoop
```

4.3. Порядок выполнения работы

Пользуясь программой 4.1 в качестве образца, выполнить индивидуальное задание в соответствии с таблицей 4.1.

Таблица 4.1 — Индивидуальное задание по лабораторной работе № 2

№ п/п	Задание
1	Число 1234 на статических семисегментных индикаторах должно светиться в течение 1 с, потом гаснуть на 1 с. Затем процесс должен повторяться.
2	В 4-м разряде статического индикатора с задержкой на 1 с число должно увеличиваться от 0 до *FFh*. После того, как число станет равным *FFh*, процесс должен повторяться сначала.
3	Цифра 5 должна перемещаться по статическому индикатору слева напра-

№ п/п	Задание
	во (остальные индикаторы должны быть погашены). Процесс должен циклически повторяться.
4	На статический индикатор выводить последовательно (с интервалом 1 с) члены геометрической прогрессии с начальным значением 1 и знаменателем 2.
5	Сделать бегущую точку слева направо на статическом семисегментном индикаторе. По достижении точкой крайней правой позиции процесс должен повторяться.
6	На статическом семисегментном индикаторе число 15*.xx* (*x* — индикатор погашен) меняться на число xx20, потом снова на 15*.xx* и т. д. Каждое из двух чисел до того, как будет изменено, должно светиться в течение 1 с.
7	Выводить по очереди на статический семисегментный индикатор числа 1111, 2222, 3333, потом процесс должен повторяться. Каждое число должно индицироваться в течение 1 с.
8	Выводить убывающие числа от *FFh* до 0 в одном из разрядов статического семисегментного индикатора, остальные разряды должны быть погашены. Каждое число до изменения должно индицироваться в течение 1 с.
9	Выводить на статический семисегментный индикатор члены геометрической прогрессии с начальным значением *FFh* и знаменателем ½.
10	Выводить перемещающееся слева направо увеличивающееся одноразрядное число от 1 до 4 по разрядам семисегментных индикаторов.
11	Поверх разрядов числа *BBBBh* слева направо должна перемещаться цифра *Ah*.
12	На левом индикаторе число *AAh* должно мигать с частотой 1 Гц, на правом индикаторе число *BBh* должно мигать с частотой 2 Гц.
13	Выводить перемещающееся циклически справа налево уменьшающееся от 9 до 6 одноразрядное число по разрядам семисегментных индикаторов.
14	Число *FFh* на левом индикаторе должно мигнуть два раза. После этого индикаторы должны быть выключены на 2 с. Этот процесс должен циклически повторяться.
15	На статических индикаторах должно поочередно отображаться шестнадцатеричное число, каждый раз увеличивающееся на единицу, начиная со значения 1. При этом каждое новое значение должно отображаться на другом индикаторе так, чтобы на левом индикаторе всегда высвечивались нечетные числа, а на правом — четные.
16	Выводить на правом индикаторе числа, увеличивающиеся на 1, начиная с 1*h* и до *Fh*. Этот процесс должен циклически повторяться.

4.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

4.5. Вопросы для самоконтроля

4.5.1. Как зажечь точку в одном разряде статического семисегментного индикатора?

4.5.2. Как погасить все разряды статического семисегментного индикатора?

4.5.3. Чем отличается статическая индикация от динамической?

4.5.4. Какой командой осуществляется увеличение содержимого регистра на единицу?

4.5.5. Можно ли выводить произвольную шестнадцатеричную цифру в один разряд статического семисегментного индикатора (при погашенных остальных разрядах)?

5. ЛАБОРАТОРНАЯ РАБОТА № 3
ПРОГРАММИРОВАНИЕ ТОЧЕЧНОГО ИНДИКАТОРА

5.1. Цель работы

Целью работы является приобретение навыков программирования на ассемблере с использованием подпрограмм.

5.2. Теоретические сведения

В отладочном стенде используется точечный индикатор, точки которого выбираются с помощью порта *A* (столбцы) и порта *C* (строки).

Адрес порта *A* — 8000*h*, адрес порта *C* — 8002*h*.

Младшие разряды порта *A* слева, младшие разряды порта *C* — сверху. При этом единицы в соответствующих разрядах порта *A* выбирают столбцы, а нули в соответствующих разрядах порта *C* выбирают строки (рис. 5.1).

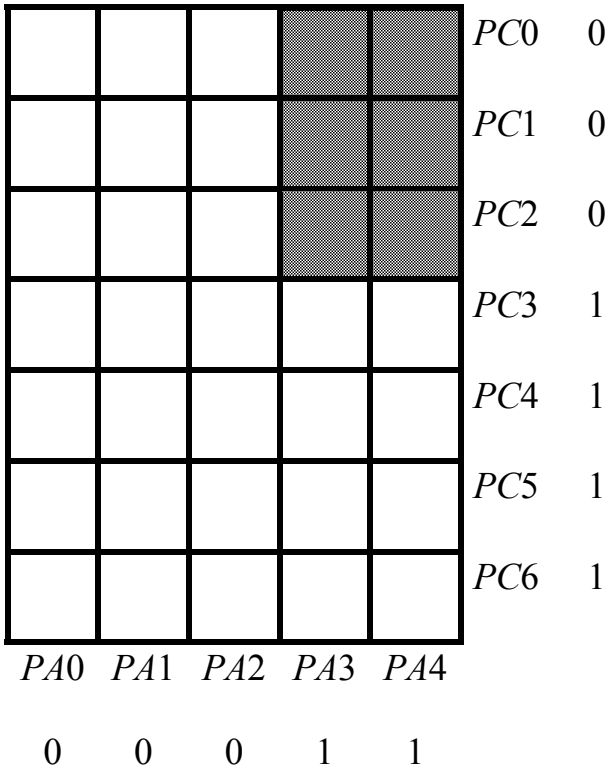


Рис. 5.1 — Точечный индикатор

Для показанного на рис. 5.1 свечения прямоугольника 3×2 в правом верхнем углу точечного индикатора необходимо вывести в порт *A* число *xxx*11000*b* и в порт *C* — число *x*1111000*b*, где разряды *x* — не имеют значения (например, *A* = 00011000*b* и *C* = 11111000*b*).

Пример программирования точечного индикатора приведен в программе 5.1.

Программа 5.1 — Программа, выводящая на точечный индикатор прямоугольник 3×2, циклически перескакивающий между правым верхним и левым

нижним углами индикатора.

```
.DEVICE AT90s8515          ; Выбираем устройство
.EQU  MCUCR = 0x35         ; Устанавливаем для регистра управления
                           микроконтроллером MCUCR адрес 0x35
.CSEG
ldi r16,0b11000000         ; В регистр управления MCUCR заносим
                           11000000b. Тем самым переводим микроконтрол-
                           лер в режим работы с внешней памятью
out MCUCR,r16
;-----------------------------------------
MainLoop:
ldi r16, 0b00011000
sts 0x8000,r16             ; Выводим в порт A число 00011000b
ldi r16, 0b11111000
sts 0x8002,r16             ; Выводим в порт C число 11111000b
                ; Теперь должен светиться правый верхний прямоугольник
ldi r19,10      ; Организуем задержку с помощью трех вложенных друг в
                друга циклов
delay3: clr r18
delay2: clr r17
delay1:
dec r17
brne delay1
dec r18
brne delay2
dec r19
brne delay3       ; Конец задержки
ldi r16, 0b00000011
sts 0x8000,r16             ; Выводим в порт A число 00000011b
ldi r16, 0b00001111
sts 0x8002,r16             ; Выводим в порт C число 00001111b
                ; Теперь должен светиться левый нижний прямоугольник
ldi r19,10      ; Организуем задержку с помощью трех вложенных друг в
                друга циклов
delay31: clr r18
delay21: clr r17
delay11:
dec r17
brne delay11
dec r18
brne delay21
dec r19
brne delay31        ; Конец задержки
rjmp MainLoop       ; Переходим на метку MainLoop.
```

Для того, чтобы можно было пользоваться подпрограммами, необходимо до начала программы разрешить работу со стеком. Как это сделать, показано в нижеприведенной программе 5.2, в которой делается то же, что и в программе 5.1, но с использованием подпрограммы.

Программа 5.2 — Программа, выводящая на точечный индикатор прямоугольник 3×2, циклически перескакивающий между левым нижним и правым верхним углами индикатора. При этом временная задержка вынесена в подпрограмму.

```
.DEVICE AT90s8515
.EQU MCUCR=0x35
.CSEG
.org 0
ldi r16, low(RAMEND)   ; начало инициализации стека
out SPL, r16
ldi r16,high(RAMEND)
out SPH, r16           ; конец инициализации стека
ldi r16, 0b11000010
out MCUCR, r16
.equ SPH=0x3E
.equ SPL=0x3D
.equ RAMEND=0x25F
;-----------------------------------------------------
MainLoop:
ldi r16, 0b00011000
sts 0x8000,r16         ;Выводим в порт A число 00011000b
ldi r16, 0b11111000
sts 0x8002,r16         ;Выводим в порт C число 11111000b
                ;Теперь должен светиться правый верхний прямоугольник
rcall delay    ; Вызов подпрограммы задержки
ldi r16, 0b00000011
sts 0x8000,r16         ;Выводим в порт A число 00000011b
ldi r16, 0b00001111
sts 0x8002,r16         ;Выводим в порт C число 00001111b
                ;Теперь должен светиться левый нижний прямоугольник
rcall delay    ; Вызов подпрограммы задержки
rjmp MainLoop          ; Переходим на метку MainLoop
; Подпрограмма задержки:
delay:
ldi r19,10     ;Организуем задержку с помощью трех вложенных друг в
                друга циклов
delay3: clr r18
delay2: clr r17
delay1:
dec r17
brne delay1
```

```
dec r18
brne delay2
dec r19
brne delay3      ; конец задержки
ret              ; команда возврата в основную программу
```

5.3. Порядок выполнения работы

Используя программу 5.1 в качестве образца, выполнить индивидуальное задание в соответствии с таблицей 5.1 (без использования подпрограмм).

Используя программу 5.2 в качестве образца, изменить программу, внеся задержку в подпрограмму.

Таблица 5.1 — Индивидуальное задание по лабораторной работе № 3

№ п/п	Задание
1	Зажигать точки на точечном индикаторе в шахматном порядке
2	В течение 1 с светится верхняя часть точечного индикатора, затем на 1 с зажигается нижняя часть точечного индикатора, потом процесс циклически повторяется.
3	Бегущая сверху вниз точка по среднему столбцу точечного индикатора. По достижении нижней позиции, повторять движение сверху.
4	Угловой квадрат, перемещающийся против часовой стрелки по точечному индикатору.
5	Зажигающиеся по очереди (по часовой стрелке) угловые точки точечного индикатора.
6	Расширяющийся левый нижний угловой квадрат (от одной точки до 5x5).
7	Перемещающиеся по диагонали точки (с левой верхней позиции до правой нижней).
8	Центральный расширяющийся квадрат (от одной центральной точки до 5x5).
9	Отобразить на точечном индикаторе букву «*A*».
10	Отобразить на точечном индикаторе вертикальный крест (светятся центральная строка и центральный столбец).
11	Постепенно, строка за строкой, зажигать точечный индикатор (сверху вниз). Циклически повторять.
12	Постепенно, столбец за столбцом, зажигать точечный индикатор (справа налево). Циклически повторять.
13	«Падающий» зигзагообразно сверху вниз квадрат 2x2. (Четыре позиции: левый верхний угол, правая сторона на 1/3 от верха, левая сторона на 1/3 от верха, левый нижний угол).
14	Пульсирующий (1 с светится, 1 с не светится) центральный квадрат 3x3.
15	Вывести букву «*O*» (светятся верхняя и нижняя стоки и правый и нижний столбцы).

№ п/п	Задание
16	Вывести букву «*S*» (светятся верхняя, средняя и нижняя строки, верхняя половина левого столбца, нижняя половина правого столбца).

5.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

5.5. Вопросы для самоконтроля

5.5.1. Какие изменения в программу нужно внести, чтобы в ней использовать подпрограмму?

5.5.2. Каким образом восстанавливается значение программного счетчика при завершении выполнения подпрограммы?

5.3.3. Какие ассемблерные команды используются для организации перехода к подпрограмме и выхода из нее?

5.3.4. Как осуществляется выбор строк и столбцов точечного индикатора?

5.3.5. Какие строки и столбцы и в каком порядке должны быть выбраны для отображения на точечном индикаторе буквы *A*?

5.3.6. Какие строки и столбцы и в каком порядке следует выбрать для отображения на точечном индикаторе шахматного поля?

6. ЛАБОРАТОРНАЯ РАБОТА № 4
ВВОД ДАННЫХ В МИКРОКОНТРОЛЛЕРНУЮ СИСТЕМУ С ПОМОЩЬЮ КЛАВИАТУРЫ

6.1. Цель работы

Целью работы является изучение программирования ввода данных в микроконтроллерную систему с помощью клавиатуры

6.2. Теоретические сведения

Матрица клавиатуры состоит из трех столбцов. Адрес левого столбца — 9006h, адрес среднего столбца — 9005*h*, адрес правого столбца — 9003*h*. По этим адресам находятся коды нажатых в соответствующем столбце клавиш. При этом каждой клавише в столбце поставлен в соответствие разряд в считываемом числе. При нажатой клавише в этот разряд записывается 0, для ненажатых клавиш в их разряды записываются единицы. Младшим разрядам соответствуют верхние клавиши, старшим — нижние. Например, для нажатой на клавиатуре цифре 7 (левый столбец, 3-я клавиша сверху) коды составят: по адресу 9006*h* — 11111011*b*, по адресу 9005*h* — 11111111*b*, по адресу 9003*h* — 11111111*b*.

Программа распознавания нажатой клавиши может быть выполнена с помощью команды вычитания *subi*, находящей разность между известным кодом клавиши и числом, полученным в результате считывания числа по адресу соответствующего столбца. Если результат вычитания не 0, то происходит возврат к началу программы и следующая проверка, если коды совпали (результат вычитания 0), то выполняется подпрограмма действий для случая обнаружения нажатия клавиши (см. рис. 6.1).

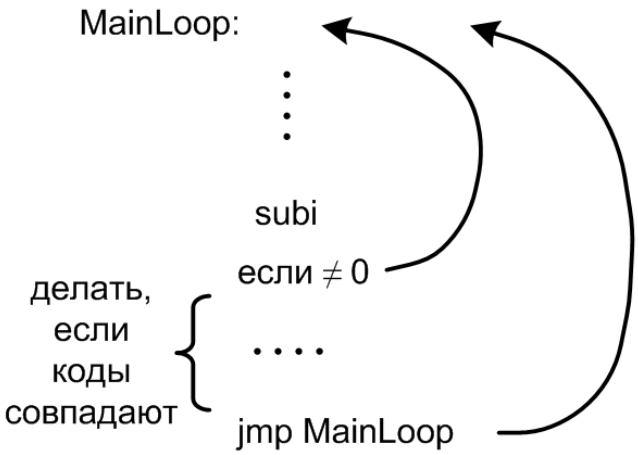


Рис. 6.1 — План построения программы обработки нажатия клавиши

Примеры программирования работы клавиатуры приведены в Программах 6.1 и 6.2.

Программа 6.1 — По короткому однократному нажатию клавиши цифры 3 на клавиатуре на время 3 с зажигаются угловые точки на точечном индикаторе.

```
.DEVICE AT90S8515
.EQU MCUCR=0x35
.CSEG
.org 0
   ldi r16,0b11000000
   out MCUCR,r16
;-------------------------------------------
   ldi r21,0b00000100
   ldi r22,0b00010001
   ldi r23,0b10111110
   ldi r24,0b00000111
   ldi r25,0b11111111
   ldi r26,0b00000000
MainLoop:
m:
  sts    0x8002, r25
  sts    0x8000, r26
  lds    r16,0x9003
  sts 0xA006, r16          ; Вывод кода нажатой клавиши на линейку свето-
                           диодов
  subi    r16, 0b11111110; Вычитание кода клавиши из принятого кода
  brne  m                  ; Если коды не совпали, перейти на метку m
  sts    0x8002, r23
  sts    0x8000, r22
  ldi     r19,11           ; Начало задержки
delay3d: clr r18
delay2d: clr r17
delay1d: dec r17
       brne delay1d
       dec r18
       brne delay2d
       dec r19
       brne delay3d        ; Конец задержки
 rjmp MainLoop
```

Программа 6.2 — По нажатию на клавиатуре клавиши цифры 5 зажигается верх точечного индикатора и светится, пока клавиша с цифрой 5 остается нажатой.

```
.DEVICE AT90S8515
.EQU MCUCR=0x35
.CSEG
.org 0
   ldi r16,0b11000000
   out MCUCR,r16
   ;-------------------------------
   ldi r21,0b00000100
```

```
    ldi r22,0b11111000
    ldi r23,0b11111111
    ldi r24,0b00000111
    ldi r25,0b11111111
    ldi r26,0b00000000
MainLoop:
m:
  sts    0x8002, r25
  sts    0x8000, r26
  lds    r16,0x9005
  sts 0xA006, r16          ; Вывод кода нажатой клавиши на линейку светодиодов
  subi   r16, 0b11111101; Вычитание кода клавиши из принятого кода
  brne  m                  ; Если коды не совпали, перейти на метку m
  sts    0x8002, r22
  sts    0x8000, r23
  ldi    r19,1             ; Начало задержки
delay3d: clr r18
delay2d: clr r17
delay1d: dec r17
       brne delay1d
       dec r18
       brne delay2d
       dec r19
       brne delay3d        ; Конец задержки
 rjmp MainLoop
```

6.3. Порядок выполнения работы

Используя программы 6.1 и 6.2 в качестве образцов, выполнить индивидуальное задание в соответствии с таблицей 6.1.

Таблица 6.1 — Индивидуальное задание по лабораторной работе № 4

№ п/п	Задание
1	При нажатии кнопок 4, 5, 6 на клавиатуре отображать их значения на статическом семисегментном индикаторе.
2	При нажатии на кнопку 5 на клавиатуре две верхние угловые точки точечного индикатора должны зажигаться по очереди циклически.
3	При нажатии кнопки 7 на клавиатуре должна зажигаться левая верхняя угловая точка, при нажатии 8 — правая нижняя угловая точка точечного индикатора.
4	При нажатии кнопок 2 или 3 на клавиатуре отображать на статическом семисегментном индикаторе значения, на 2 большие (соответственно 4 и 5).
5	При нажатии кнопки 1 на клавиатуре зажигать весь точечный индикатор,

№ п/п	Задание
	при нажатии на 4 — гасить.
6	При нажатии кнопки 9 на клавиатуре зажигать весь точечный индикатор на время 1 с. После этого точечный индикатор должен погаснуть.
7	Каждое нажатие на кнопку 4 клавиатуры должно сдвигать светящуюся строку точечного индикатора на одну позицию вниз, после достижения последней позиции процесс должен повторяться сначала.
8	Каждое нажатие на кнопку 9 клавиатуры должно смещать точку в семисегментном индикаторе на один разряд вправо.
9	На время удержания нажатой кнопки 2 на клавиатуре значение на семисегментном индикаторе должно увеличиваться на 1 с задержкой на 1 с после каждого увеличения значения.
10	При однократном нажатии кнопки 7 на клавиатуре центральная точка точечного индикатора должна мигнуть 3 раза.
11	После одновременном нажатии кнопок 2 и 4 на клавиатуре должны попеременно циклически зажигаться верхняя и нижняя половины точечного индикатора.
12	После нажатия кнопки 8 на клавиатуре на точечном индикаторе должен два раза мигнуть центральный квадрат 2x2.
13	После нажатия на кнопку 9 на клавиатуре на точечном индикаторе должны попеременно циклически зажигаться угловые квадраты 3x3, расположенные на главной диагонали точечного индикатора.
14	При нажатии кнопки 5 должна зажечься на время 3 с верхняя половина точечного индикатора, при нажатии кнопки 1 — должна зажечься на время 3 с нижняя половина точечного индикатора.
15	На время удержания нажатой кнопки 6 клавиатуры должен мигать крайний левый столбец точечного индикатора.
16	После нажатия кнопки 3 на клавиатуре должны циклически зажигаться левая и правая половины точечного индикатора.

6.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

6.5. Вопросы для самоконтроля

6.5.1. Как нужно изменить программу 6.1, чтобы программа выполнила действие при одновременном нажатии двух определенных клавиш?

6.5.2. Как осуществляется проверка столбца клавиатуры на нажатие клавиши?

6.5.3. Как производится идентификация клавиши в пределах одного столбца клавиатуры?

6.5.4. Как организовать однократное выполнение какого-нибудь события при однократном кратковременном нажатии клавиши клавиатуры?

6.5.5. Как организовать циклическое выполнение события на время удер-

жания клавиши нажатой?

6.5.6. Как организовать выполнение события заданное число раз при кратковременном однократном нажатии клавиши

7. ЛАБОРАТОРНАЯ РАБОТА № 5
ПРОГРАММИРОВАНИЕ ВНЕШНИХ ПРЕРЫВАНИЙ

7.1. Цель работы

Целью работы является изучение программирования обработки внешних прерываний.

7.2. Теоретические сведения

Начальные ячейки программы обычно отводятся под векторы прерываний (команды перехода с адресом начала подпрограммы обработки прерывания). Описание векторов прерываний приведено в таблице 7.1.

Таблица 7.1 — Векторы прерываний

№ п/п	Адрес вектора	Описание вектора прерывания
1	000*h*	Старт программы после сброса, включения питания или сброса по сторожевому таймеру
2	001*h*	Внешнее прерывание 0 (*pin INT*0)
3	002*h*	Внешнее прерывание 1 (*pin INT*1)
4	003*h*	Таймер/счетчик 1 — захват
5	004*h*	Таймер/счетчик 1 — сравнение *A*
6	005*h*	Таймер/счетчик 1 — сравнение *B*
7	006*h*	Таймер/счетчик 1 — переполнение
8	007*h*	Таймер/счетчик 0 — переполнение
9	008*h*	Прерывание *SPI* — цикл обмена завершен
10	009*h*	Прерывание *USART* — принят байт
11	00*Ah*	Прерывание *USART* — регистр данных пуст
12	00*Bh*	Прерывание *USART* — передача закончена
13	00*Ch*	Аналоговый компаратор

В соответствии с таблицей 7.1 для обработки внешних прерываний могут быть использованы два вектора прерываний с адресами 001*h* и 002*h*. Если прерывания будут использоваться, то до начала работы программы в ней должны быть записаны подпрограммы обработки внешних прерываний.

После включения питания микроконтроллер приступает к выполнению основной программы, начиная с нулевой ячейки. При поступлении сигнала единичного уровня на вход *INT*0 микроконтроллера (при нажатии кнопки *INT*0 на отладочном стенде, см. Приложение Б) управление будет передано ячейке памяти программ с номером 001*h*. В этой ячейке находится команда перехода к началу подпрограммы обработки прерывания. Микроконтроллер закончит выполнение текущей команды основной программы, запомнит текущее состояние счетчика команд в стеке, и перейдет к выполнению подпрограммы обработки прерывания. По команде *reti* выполнение подпрограммы обработки прерывания заканчивается, микроконтроллер восстанавливает из стека значение программ-

ного счетчика, и программа будет выполняться с того места, где была прервана.

Аналогично обрабатывается запрос прерывания при поступлении сигнала единичного уровня на вход *INT*1 микроконтроллера (при нажатии кнопки *INT*1 на отладочном стенде, см. Приложение Б). При этом управление будет передано ячейке памяти программ с номером 002*h*, содержащей вектор прерываний.

Пример работы с внешним прерыванием приведен в нижеследующей программе 7.1. Структура программы:

— директивы препроцессора;
— область векторов прерываний;
— подпрограмма обработки прерывания;
— область инициализации;
— основная программа.

Программа 7.1 — По нажатию кнопки внешнего прерывания *INT*0 прерывается выполнение основной программы,

```
.DEVICE AT90s8515
.EQU MCUCR=0x35
.EQU GIFR=0x3A
.EQU GICR=0x3B
.CSEG
.org 0
.equ SPH=0x3E
.equ SPL=0x3D
.equ RAMEND=0x25F
;----------------------------------------------------
rjmp Reset          ; Переход к основной программе
rjmp INT_0          ; Переход к подпрограмме обработки прерывания
nop  ;rjmp INT_1
nop  ;rjmp Timer1_capt1
nop  ;rjmp Timer1_compA
nop  ;rjmp Timer1_compB
nop  ;rjmp Timer1_OVF1
nop  ;rjmp Timer0_OVF0
nop  ;rjmp SPI_STC
nop  ;rjmp UART_RX
nop  ;rjmp UART_UDRE
nop  ;rjmp UART_TX
nop  ;rjmp ANA_COMP
INT_0:              ; Начало подпрограммы обработки прерывания
 ldi    r16,0b01010101
        sts    0xA006, r16
 rcall p
 reti               ; Конец подпрограммы обработки прерывания

Reset:              ; Начало основной программы
;-------- начало инициализации  -----
```

```
 ldi r16, low(RAMEND)          ; начало инициализации стека
 out SPL, r16
 ldi r16,high(RAMEND)
 out SPH, r16                  ; конец инициализации стека


   ldi r16, 0b11000010
   out MCUCR,r16

   ldi r16, 0x40
   out GICR,r16

   ldi r16, 0x40
   out GIFR,r16

    sei          ; разрешить прерывания, cli — запрещение прерываний
;-------- конец инициализации  -----

   ldi r23,0b11111111
   ldi r24,0b00000111
   ldi r25,0b11111100
   ldi r26,0b00000001

MainLoop:
 cli                           ; Запретить прерывания
 sts    0x8002, r25            ; Зажечь две точки в левом верхнем углу
 sts    0x8000, r26
    rcall p             ; Перейти к подпрограмме задержки

 sts    0x8002, r23            ; Погасить точечный индикатор
 sts    0x8000, r24
   rcall p                     ; Перейти к подпрограмме задержки
 sei                           ; Разрешить прерывания

 rjmp MainLoop                 ; Прейти к началу основной программы

p:                             ; Начало подпрограммы задержки
        ldi r19,50
delay3: clr r18
delay2: clr r17
delay1: dec r17
       brne delay1
       dec r18
       brne delay2
       dec r19
```

```
    brne delay3
ret                         ; Конец подпрограммы задержки.
```

7.3. Порядок выполнения работы

Используя программу 7.1 в качестве образца, выполнить индивидуальное задание в соответствии с таблицей 7.2.

Таблица 7.2 — Индивидуальное задание по лабораторной работе № 5

№ п/п	Задание
1	Основная программа: мигает цифра 7 на статическом семисегментном индикаторе (с частотой 1 Гц). Подпрограмма обработки прерывания: три раза зажигается точечный индикатор (каждый раз на время 1 с с интервалом в 1 с).
2	Основная программа: мигает точечный индикатор (все точки точечного индикатора зажигаются на время 1 с и гаснут на время 1 с). Подпрограмма обработки прерывания: На точный индикатор на время 3 с выводится светящаяся вертикальная линия посередине.
3	Основная программа: на статическом семисегментном индикаторе с интервалом времени в 1 с бежит слева направо цифра 5. Подпрограмма обработки прерывания: на статический семисегментный индикатор на время 3 с выводится число 8888.
4	Основная программа: на точечном индикаторе светится прямой крест. Подпрограмма обработки прерывания: на точечном индикаторе два раза с интервалом времени в 1 с мигает центральный квадрат размером 3x3.
5	Основная программа: на статическом семисегментном индикаторе мигает число 1234 с интервалом времени в 1 с. Подпрограмма обработки прерывания: на точечном индикаторе на время 2 с зажигаются все угловые точки.
6	Основная программа: бегущий слева направо огонек на линейке светодиодов. Подпрограмма обработки прерывания: на время 4 с зажигается нижняя половина точечного индикатора.
7	Основная программа: на точечном индикаторе вращающийся против часовой стрелки угловой квадрат размером 2x2. Подпрограмма обработки прерывания: 3 раза с интервалом времени в 1 с зажигается весь точечный индикатор.
8	Основная программа: на точечном индикаторе зажигаются попеременно центральная точка и центральные квадрат размером 3x3. Подпрограмма обработки прерывания: на одном из семисегментных индикаторов два раза мигает число *AAh*.
9	Основная программа: на точечном индикаторе по очереди зажигаются угловые точки, расположенные на побочной диагонали точечного индикатора.

№ п/п	Задание
	Подпрограмма обработки прерывания: на семисегментном индикаторе три раза зажигается и гаснет число 1*A2Bh*.
10	Основная программа: на точечном индикаторе поочередно зажигаются центральный вертикальный столбец и центральная горизонтальная строка. Подпрограмма обработки прерывания: на точечном индикаторе на время 3 с зажигается центральная точка.
11	Основная программа: на точечном индикаторе мигает центральная точка. Подпрограмма обработки прерывания: на семисегментном индикаторе на время 2 с зажигается число *DDDDh.*
12	Основная программа: на семисегментном индикаторе поочередно меняются числа 1010*h* и 0505*h*. Подпрограмма обработки прерывания: три раза зажигаются и гаснут угловые точки на точечном индикаторе.
13	Основная программа: на семисегментном индикаторе циклически зажигаются 77*h* на левом индикаторе (правый в это время погашен) и число *CCh* на правом индикаторе (в это время левый индикатор погашен). Подпрограмма обработки прерывания: по часовой стрелке однократно последовательно зажигаются угловые точки точечного индикатора.
14	Основная программа: на левом семисегментном индикаторе (правый погашен) отображается увеличивающееся число от 0 до FFh с шагом 1, затем процесс циклически повторяется. Подпрограмма обработки прерывания: на правом семисегментном индикаторе на время 5 с зажигается число 37*h*.
15	Основная программа: на правом семисегментном индикаторе по очереди циклически отображаются числа 21*h* и *ACh*. Левый индикатор погашен. Подпрограмма обработки прерывания: На левом индикаторе два раза мигает число *EEh* (появляется на время 1 с и гаснет на время 1 с).
16	Основная программа: на семисегментном индикаторе светится *x*7*x*7 (*x* — разряд погашен), на точечном индикаторе — центральный квадрат размером 5x5. Подпрограмма обработки прерывания: на время 3 с на семисегментном индикаторе зажигается число *DE*32*h*, а на точечном индикаторе в это время светится центральная точка.

7.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

7.5. Вопросы для самоконтроля

7.5.1. Сколько внешних прерываний может быть в микроконтроллере *ATMega*8515?

7.5.2. Дайте определение вектору прерывания.

7.5.3. В чем заключается отличие в выполнении команд завершения подпрограммы *RET* и завершения подпрограммы обработки прерывания *RETI*?

7.5.4. Что такое прерывание?

7.5.5. Чем отличаются внешние прерывания от внутренних прерываний?

7.5.6. Как программно разрешить или запретить выполнение прерываний?

7.5.7. Где хранятся векторы прерываний?

7.5.8. Почему для обработки прерываний должен быть инициализирован стек?

8. ЛАБОРАТОРНАЯ РАБОТА № 6
ПРОГРАММИРОВАНИЕ ТАЙМЕРА СЧЕТЧИКА

8.1. Цель работы

Целью работы является приобретение навыков составления программы обработки внутренних прерываний

8.2. Теоретические сведения

В состав микроконтроллера *ATMega*8515 входят два таймера/счетчика: восьмиразрядный таймер/счетчик *T*/*C*0 и шестнадцатиразрядный таймер/счетчик *T*/*C*1. Эти таймеры/счетчики суммирующие. При работе в качестве счетчика содержимое регистра таймера/счетчика увеличивается на единицу при приходе импульса на соответствующий вход микроконтроллера (*T*0 или *T*1). При работе в качестве таймера содержимое регистра таймера/счетчика увеличивается на каждый *n*-й импульс тактового генератора (коэффициент деления *n* задается при программировании таймера/счетчика). При переполнении регистра таймера/счетчика вырабатывается запрос на прерывание.

Рассмотрим, как формируется с помощью таймера/счетчика *T*/*C*1 временной интервал, равный 1 с. При тактовой частоте f = 4 МГц и коэффициенте деления частоты 1024 содержимое регистра таймера/счетчика будет увеличиваться на единицу каждые 1024/4000000 = 0,000256 с. Для формирования интервала времени в 1 с таких изменений значения таймера/счетчика должно быть 1/0,000256 = 3906. Переполнение происходит, когда содержимое шестнадцатиразрядного регистра таймера/счетчика достигает значения *FFFFh* + 1 = 65536. Таким образом, для формирования интервала в 1 с перед началом счета в регистр *T*/*C*1 надо записать число 65536 – 3906 = 61630. Шестнадцатиразрядный регистр *T*/*C*1 обозначается *TCNT*1 и состоит из восьмиразрядных старшей *TCNT*1*H* и младшей *TCNT*1*L* частей.

Подробнее с режимами работы и программированием таймеров/счетчиков можно ознакомиться в литературе [2..5], приведенной в библиографическом списке.

Программа 8.1 — В основной программе зажигаются все светодиоды линейки светодиодных индикаторов, после этого основная программа зацикливается. По истечении интервала времени 1 с, отмеряемом таймером/счетчиком1, инвертируются все разряды числа, управляющего свечением светодиодов. Далее таймер/счетчик1 снова отмеряет 1 с, после чего производится инверсия и т. д.

```
.DEVICE AT90s8515
.EQU MCUCR=0x35
.EQU GIFR=0x3A
.EQU GICR=0x3B
.CSEG
.org 0
.equ SPH=0x3E
```

```
.equ SPL=0x3D
.equ RAMEND=0x25F
.equ TCCR1A=0x2F
.equ TCCR1B=0x2E
.equ TCNT1L=0x2C
.equ TCNT1H=0x2D
.equ TIFR=0x38
.equ TIMSK=0x39
.equ GIMSK=0x3B

rjmp Reset                    ; Переход к основной программе
nop  ;rjmp INT_0
nop  ;rjmp INT_1
nop  ;rjmp Timer1_capt1
nop  ;rjmp Timer1_compA
nop  ;rjmp Timer1_compB
rjmp Timer1_OVF1 ; Переход к подпрограмме по переполнению T/C1
nop  ;rjmp Timer0_OVF0
nop  ;rjmp SPI_STC
nop  ;rjmp UART_RX
nop  ;rjmp UART_UDRE
nop  ;rjmp UART_TX
nop  ;rjmp ANA_COMP

Timer1_OVF1:          ; Начало подпрограммы обработки прерывания по
                      переполнению таймера/счетчика1
 com  r17 ; инверсия
      sts    0xA006, r17

 ldi r16, 0b10111110
 out TCNT1L,r16       ; Помещаем число 61630 в регистр T/C1
 ldi r16,0b11110000
 out TCNT1H,r16

 reti                 ; Конец подпрограммы обработки прерывания по
                      переполнению таймера/счетчика1
;------- начало инициализации  -----
Reset:
 ldi r16, low(RAMEND)      ; начало инициализации стека
 out SPL, r16
 ldi r16,high(RAMEND)
 out SPH, r16              ; конец инициализации стека

  ldi r16, 0b11000000
  out MCUCR,r16

  ;ldi r16, 0x40
```

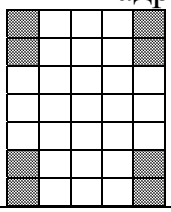
```
  ;out GICR,r16

  ;ldi r16, 0x40
  ;out GIFR,r16

 ldi r16, 0b10111110
 out TCNT1L,r16          ;61630
 ldi r16, 0b11110000
 out TCNT1H,r16

 ldi r16, 0
 out TCCR1A, r16 ; ШИМ не активирован
 ldi r16, 5
 out TCCR1B, r16 ; коэффициент деления частоты 1024

 ldi r16, 0
 out TIFR, r16  ; сброс флага переполнения T/C1
 ldi r16, 0x80
 out TIMSK, r16 ; разрешено прерывание по переполнению T/C1
 ldi r16, 0
 out GIMSK, r16 ; внешнее прерывание запрещено
 sei ; разрешить прерывания, cli — запрещение прерываний по умолчанию
;-------- конец инициализации  -----

 ldi     r17,0b11111111
         sts     0xA006, r17 ; зажечь все светодиоды

MainLoop:
 m1: jmp m1
 rjmp MainLoop
```

8.3. Порядок выполнения работы

Пользуясь программой 8.1 в качестве образца, выполнить индивидуальное задание в соответствии с таблицей 8.1.

Таблица 8.1 — Индивидуальное задание по лабораторной работе № 6

№ п/п	Задание
1	На точечном индикаторе светится центральная точка. Через каждые 3 с на время 3 с зажигается число 7*h* на семисегментном индикаторе.
2	Зажигаются по очереди на время 2 с верхняя и нижняя половины точечного индикатора.
3	Каждые 3 с зажигаются угловые точки на точечном индикаторе и на время 3 с гаснут. Затем процесс циклически повторяется.

№ п/п	Задание
4	На статическом семисегментном индикаторе светится число 1234. На время 2 с зажигается точечный индикатор, затем на время 2 с гаснет. После этого процесс повторяется.
5	На точечном индикаторе по очереди на время 3 с зажигаются центральный квадрат размером 3x3 и угловые вертикальные отрезки:  .
6	На точечном индикаторе светятся верхняя и нижняя строки. На статическом семисегментном индикаторе на время 5 с зажигается *x*50*x* и гаснет на время 5 с. Затем процесс циклически повторяется.
7	На статическом семисегметном индикаторе светится число 5555. Каждые 3 с зажигается центральный квадрат 3x3 и гаснет на время 3 с. Затем процесс циклически повторяется.
8	На статическом семисегментром индикаторе в течение 1 с светится число 1234, точечный индикатор погашен. Следующую 1 с светится весь точечный индикатор, статический семисегментный индикатор гаснет. Затем процесс повторяется.
9	В течение 2 с светится линейка светодиодов (через один светодиод), статические семисегментные индикаторы погашены. Потом линейка светодиодов гаснет и в течение 2 с светится число 4444 на статических семисегментных индикаторах. Затем процесс циклически повторяется.
10	На точечном индикаторе в течение светится 1 с центральный квадрат размером 3x3, потом гаснет на время 1 с. Потом процесс циклически повторяется.
11	На линейке светодиодов циклически меняются состояния *x*x** и **x*x* с интервалом в 2 с.
12	На левом и правом семисегментном индикаторах поочередно с интервалом 1 с выводится число 45*h* (другой индикатор в это время погашен).
13	С интервалом в 2 с сменяются две ситуации: на семисегментном индикаторе светится x77x, точечный индикатор погашен; на точечном индикаторе светятся четыре угловых квадрата размером 2x2, семисегментный индикатор погашен.
14	На линейке светодиодов перемещается светящийся огонек со скоростью одна позиция за 1 с.
15	С интервалом в 1 с сменяются две ситуации: на точечном индикаторе светятся угловые квадраты размером 2x2, расположенные на главной диагонали; на точечном индикаторе светятся угловые квадраты размером 2x2, расположенные на побочной диагонали.

№ п/п	Задание
16	Зажигать на линейке светодиодных индикаторов циклически четыре левых светодиода и четыре правых светодиода.

8.4. Содержание отчета

Отчет по работе оформляется в соответствии с требованиями, приведенными во введении.

8.5. Вопросы для самоконтроля

8.5.1. Сколько и каких таймеров/счетчиков входит в состав микроконтроллера *ATMega*8515?

8.5.2. Как рассчитать начальное значение таймера/счетчика по заданному времени, которое должен отмерить таймер/счетчик до своего переполнения?

8.5.3. В чем состоит отличие работы таймера/счетчика при в режиме таймера и в режиме счетчика?

8.5.4. Для чего используется сторожевой таймер?

8.5.5. Где находятся регистры счетчиков?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Методические указания по оформлению текстовых работ для студентов дневной и заочной форм обучения направления 6.0907 — “Радиотехника” / В. Г. Слезкин. — Севастополь: Изд-во СевНТУ, 2010. — 15 с.

2. Евстифеев А.В. Микроконтроллеры *AVR* семейства Mega. Руководство пользователя / А.В. Евстифеев. — М.: Издательский дом «Додэка-XXI», 2007. — 592 с.

3. Трампет В. *AVR-RISC* микроконтроллеры. Архитектура, аппаратные ресурсы, система команд, программирование, применение / В. Трампет. — К.: МК-Пресс, 2006. — 464 с.

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

5. Сайт фирмы *ATMEL* [Электронный ресурс]. — Режим доступа: *http*://*www.atmel.com* Вторник, 28 декабря 2010 22:58:11.

ПРИЛОЖЕНИЕ А
Список ассемблерных команд *AVR* микроконтроллеров

Таблица А.1 — Арифметические и логические инструкции

Мнемоника	Операнды	Описание	Операция	Флаги	Циклы
ADD	Rd,Rr	Суммирование без переноса	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	Суммирование с переносом	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
SUB	Rd,Rr	Вычитание без переноса	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	Вычитание константы	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	Вычитание с переносом	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	Вычитание константы с переносом	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	Логическое И	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	Логическое И с константой	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	Логическое ИЛИ	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	Логическое ИЛИ с константой	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	Логическое исключающее ИЛИ	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	Побитная Инверсия	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	Изменение знака (Доп. код)	Rd = $00 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	Установить бит (биты) в регистре	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	Сбросить бит (биты) в регистре	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	Инкрементировать значение регистра	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	Декрементировать значение регистра	Rd = Rd -1	Z,N,V,S	1
TST	Rd	Проверка на ноль либо отрицательность	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	Очистить регистр	Rd = 0	Z,C,N,V,S	1
SER	Rd	Установить регистр	Rd = $FF	None	1
ADIW	Rdl,K6	Сложить константу и слово	Rdh:Rdl = Rdh:Rdl + K6	Z,C,N,V,S	2

Мнемоника	Операнды	Описание	Операция	Флаги	Циклы
SBIW	Rdl,K 6	Вычесть константу из слова	Rdh:Rdl = Rdh:Rdl - K 6	Z,C,N,V,S	2
MUL	Rd,Rr	Умножение чисел без знака	R1:R0 = Rd * Rr	Z,C	2
MULS	Rd,Rr	Умножение чисел со знаком	R1:R0 = Rd * Rr	Z,C	2
MULSU	Rd,Rr	Умножение числа со знаком с числом без знака	R1:R0 = Rd * Rr	Z,C	2
FMUL	Rd,Rr	Умножение дробных чисел без знака	R1:R0 = (Rd * Rr) << 1	Z,C	2
FMULS	Rd,Rr	Умножение дробных чисел со знаком	R1:R0 = (Rd *Rr) << 1	Z,C	2
FMULSU	Rd,Rr	Умножение дробного числа со знаком с числом без знака	R1:R0 = (Rd * Rr) << 1	Z,C	2

Таблица А.2 — Инструкции ветвления

Мнемоника	Операнды	Описание	Операция	Флаги	Циклы
RJMP	k	Относительный переход	PC = PC + k +1	None	2
IJMP	Нет	Косвенный переход на (Z)	PC = Z	None	2
EIJMP	Нет	Расширенный косвенный переход на (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND	None	2
JMP	k	Переход	PC = k	None	3
RCALL	k	Относительный вызов подпрограммы	STACK = PC+1, PC = PC + k + 1	None	3/4*
ICALL	Нет	Косвенный вызов (Z)	STACK = PC+1, PC = Z	None	3/4*
EICALL	Нет	Расширенный косвенный вызов (Z)	STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND	None	4*

Мнемо-ника	Опе-ранды	Описание	Операция	Флаги	Циклы
CALL	k	Вызов подпрограм-мы	STACK = PC+2, PC = k	None	4/5*
RET	Нет	Возврат из подпро-граммы	PC = STACK	None	4/5*
RETI	Нет	Возврат из прерыва-ния	PC = STACK	I	4/5*
CPSE	Rd,Rr	Сравнить, пропус-тить если равны	if (Rd ==Rr) PC = PC 2 or 3	None	1/2/3
CP	Rd,Rr	Сравнить	Rd -Rr	Z,C,N,V,H, S	1
CPC	Rd,Rr	Сравнить с перено-сом	Rd - Rr - C	Z,C,N,V,H, S	1
CPI	Rd,K8	Сравнить с констан-той	Rd - K	Z,C,N,V,H, S	1
SBRC	Rr,b	Пропустить если бит в регистре очищен	if(Rr(b)==0) PC = PC + 2 or 3	None	1/2/3
SBRS	Rr,b	Пропустить если бит в регистре установ-лен	if(Rr(b)==1) PC = PC + 2 or 3	None	1/2/3
SBIC	P,b	Пропустить если бит в порту очищен	if(I/O(P,b)==0) PC = PC + 2 or 3	None	1/2/3
SBIS	P,b	Пропустить если бит в порту установлен	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	Перейти если флаг в SREG очищен	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	Перейти если флаг в SREG установлен	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	Перейти если равно	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k	Перейти если не равно	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k	Перейти если пере-нос установлен	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k	Перейти если пере-нос очищен	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k	Перейти если равно или больше	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k	Перейти если мень-ше	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k	Перейти если минус	if(N==1) PC = PC + k + 1	None	1/2

Мнемоника	Операнды	Описание	Операция	Флаги	Циклы
BRPL	k	Перейти если плюс	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k	Перейти если больше или равно (со знаком)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k	Перейти если меньше (со знаком)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k	Перейти если флаг внутреннего переноса установлен	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k	Перейти если флаг внутреннего переноса очищен	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k	Перейти если флаг T установлен	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k	Перейти если флаг T очищен	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k	Перейти если флаг переполнения установлен	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k	Перейти если флаг переполнения очищен	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k	Перейти если прерывания разрешены	if(I==1) PC = PC + k + 1	None	1/2
BRID	k	Перейти если прерывания запрещены	if(I==0) PC = PC + k + 1	None	1/2

Примечание. Для операций доступа к данным количество циклов указано при условии работы с внутренней памятью данных.

Таблица А.3 — Инструкции передачи данных

Мнемоника	Операнды	Описание	Операция	Флаги	Циклы
MOV	Rd,Rr	Скопировать регистр	Rd = Rr	None	1
MOVW	Rd,Rr	Скопировать пару регистров	Rd+1:Rd = Rr+1:Rr, r,d even	None	1
LDI	Rd,K8	Загрузить константу	Rd = K	None	1
LDS	Rd,k	Прямая загрузка	Rd = (k)	None	2*
LD	Rd,X	Косвенная загрузка	Rd = (X)	None	2*
LD	Rd,X+	Косвенная загрузка с пост-инкрементом	Rd = (X), X=X+1	None	2*

LD	Rd,-X	Косвенная загрузка с пре-декрементом	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	Косвенная загрузка	Rd = (Y)	None	2*
LD	Rd,Y+	Косвенная загрузка с пост-инкрементом	Rd = (Y), Y=Y+1	None	2*
LD	Rd,-Y	Косвенная загрузка с пре-декрементом	Y=Y-1, Rd = (Y)	None	2*
LDD	Rd,Y+q	Косвенная загрузка с замещением	Rd = (Y+q)	None	2*
LD	Rd,Z	Косвенная загрузка	Rd = (Z)	None	2*
LD	Rd,Z+	Косвенная загрузка с пост-инкрементом	Rd = (Z), Z=Z+1	None	2*
LD	Rd,-Z	Косвенная загрузка с пре-декрементом	Z=Z-1, Rd = (Z)	None	2*
LDD	Rd,Z+q	Косвенная загрузка с замещением	Rd = (Z+q)	None	2*
STS	k,Rr	Прямое сохранение	(k) = Rr	None	2*
ST	X,Rr	Косвенное сохранение	(X) = Rr	None	2*
ST	X+,Rr	Косвенное сохранение с пост-инкрементом	(X) = Rr, X=X+1	None	2*
ST	-X,Rr	Косвенное сохранение с пре-декрементом	X=X-1, (X)=Rr	None	2*
ST	Y,Rr	Косвенное сохранение	(Y) = Rr	None	2*
ST	Y+,Rr	Косвенное сохранение с пост-инкрементом	(Y) = Rr, Y=Y+1	None	2
ST	-Y,Rr	Косвенное сохранение с пре-декрементом	Y=Y-1, (Y) = Rr	None	2
ST	Y+q,Rr	Косвенное сохранение с замещением	(Y+q) = Rr	None	2
ST	Z,Rr	Косвенное сохранение	(Z) = Rr	None	2
ST	Z+,Rr	Косвенное сохранение с пост-инкрементом	(Z) = Rr, Z=Z+1	None	2
ST	-Z,Rr	Косвенное сохранение с пре-декрементом	Z=Z-1, (Z) = Rr	None	2
ST	Z+q,Rr	Косвенное сохранение с замещением	(Z+q) = Rr	None	2
LPM	Нет	Загрузка из программной памяти	R0 = (Z)	None	3
LPM	Rd,Z	Загрузка из программной памяти	Rd = (Z)	None	3
LPM	Rd,Z+	Загрузка из программной памяти с пост-инкрементом	Rd = (Z), Z=Z+1	None	3

ELPM	Нет	Расширенная загрузка из программной памяти	R0 = (RAMPZ:Z)	None	3
ELPM	Rd,Z	Расширенная загрузка из программной памяти	Rd = (RAMPZ:Z)	None	3
ELPM	Rd,Z+	Расширенная загрузка из программной памяти с пост-инкрементом	Rd = (RAMPZ:Z), Z = Z+1	None	3
SPM	Нет	Сохранение в программной памяти	(Z) = R1:R0	None	-
ESPM	Нет	Расширенное сохранение в программной памяти	(RAMPZ:Z) = R1:R0	None	-
IN	Rd,P	Чтение порта	Rd = P	None	1
OUT	P,Rr	Запись в порт	P = Rr	None	1
PUSH	Rr	Занесение регистра в стек	STACK = Rr	None	2
POP	Rd	Извлечение регистра из стека	Rd = STACK	None	2

Примечание. Для операций доступа к данным количество циклов указано при условии работы с внутренней памятью данных.

Таблица А.5 — Инструкции работы с битами

Мнемоника	**Операнды**	**Описание**	**Операция**	**Флаги**	**Циклы**
LSL	Rd	Логический сдвиг влево	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	Z,C,N,V, H,S	1
LSR	Rd	Логический сдвиг вправо	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	1
ROL	Rd	Циклический сдвиг влево через C	Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7)	Z,C,N,V, H,S	1
ROR	Rd	Циклический сдвиг вправо через C	Rd(7)=C, Rd(n)=Rd(n+1), C=Rd(0)	Z,C,N,V,S	1
ASR	Rd	Арифметический сдвиг вправо	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	1
SWAP	Rd	Перестановка тетрад	Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0)	None	1
BSET	s	Установка флага	SREG(s) = 1	SREG(s)	1
BCLR	s	Очистка флага	SREG(s) = 0	SREG(s)	1

Мне-моника	Опе-ранды	Описание	Операция	Флаги	Циклы
SBI	P,b	Установить бит в порту	I/O(P,b) = 1	None	2
CBI	P,b	Очистить бит в порту	I/O(P,b) = 0	None	2
BST	Rr,b	Сохранить бит из регистра в T	T = Rr(b)	T	1
BLD	Rd,b	Загрузить бит из T в регистр	Rd(b) = T	None	1
SEC	Нет	Установить флаг переноса	C =1	C	1
CLC	Нет	Очистить флаг переноса	C = 0	C	1
SEN	Нет	Установить флаг отрицательного числа	N = 1	N	1
CLN	Нет	Очистить флаг отрицательного числа	N = 0	N	1
SEZ	Нет	Установить флаг нуля	Z = 1	Z	1
CLZ	Нет	Очистить флаг нуля	Z = 0	Z	1
SEI	Нет	Установить флаг прерываний	I = 1	I	1
CLI	Нет	Очистить флаг прерываний	I = 0	I	1
SES	Нет	Установить флаг числа со знаком	S = 1	S	1
CLN	Нет	Очистить флаг числа со знаком	S = 0	S	1
SEV	Нет	Установить флаг переполнения	V = 1	V	1
CLV	Нет	Очистить флаг переполнения	V = 0	V	1
SET	Нет	Установить флаг T	T = 1	T	1
CLT	Нет	Очистить флаг T	T = 0	T	1
SEH	Нет	Установить флаг внутреннего переноса	H = 1	H	1
CLH	Нет	Очистить флаг внутреннего переноса	H = 0	H	1
NOP	Нет	Нет операции	Нет	None	1
SLEEP	Нет	Спать (уменьшить энергопотребление)	Смотрите описание инструкции	None	1

Мне-моника	Опе-ранды	Описание	Операция	Флаги	Циклы
WDR	Нет	Сброс сторожевого таймера	Смотрите описание инструкции	None	1

Примечание. В программе на ассемблере регистр, а котором набран символ, не имеет значения (т.е. прописные и строчные буквы воспринимаются одинаково).

Обозначения, используемые для операндов:

— *Rd* — результирующий (и исходный) регистр в регистровом файле

— *Rr* — исходный регистр в регистровом файле

— *b* — константа (3 бита), может быть константное выражение

— *s* — константа (3 бита), может быть константное выражение

— *P* — константа (5-6 бит), может быть константное выражение

— *K*6 — константа (6 бит), может быть константное выражение

— *K*8 — константа (8 бит), может быть константное выражение

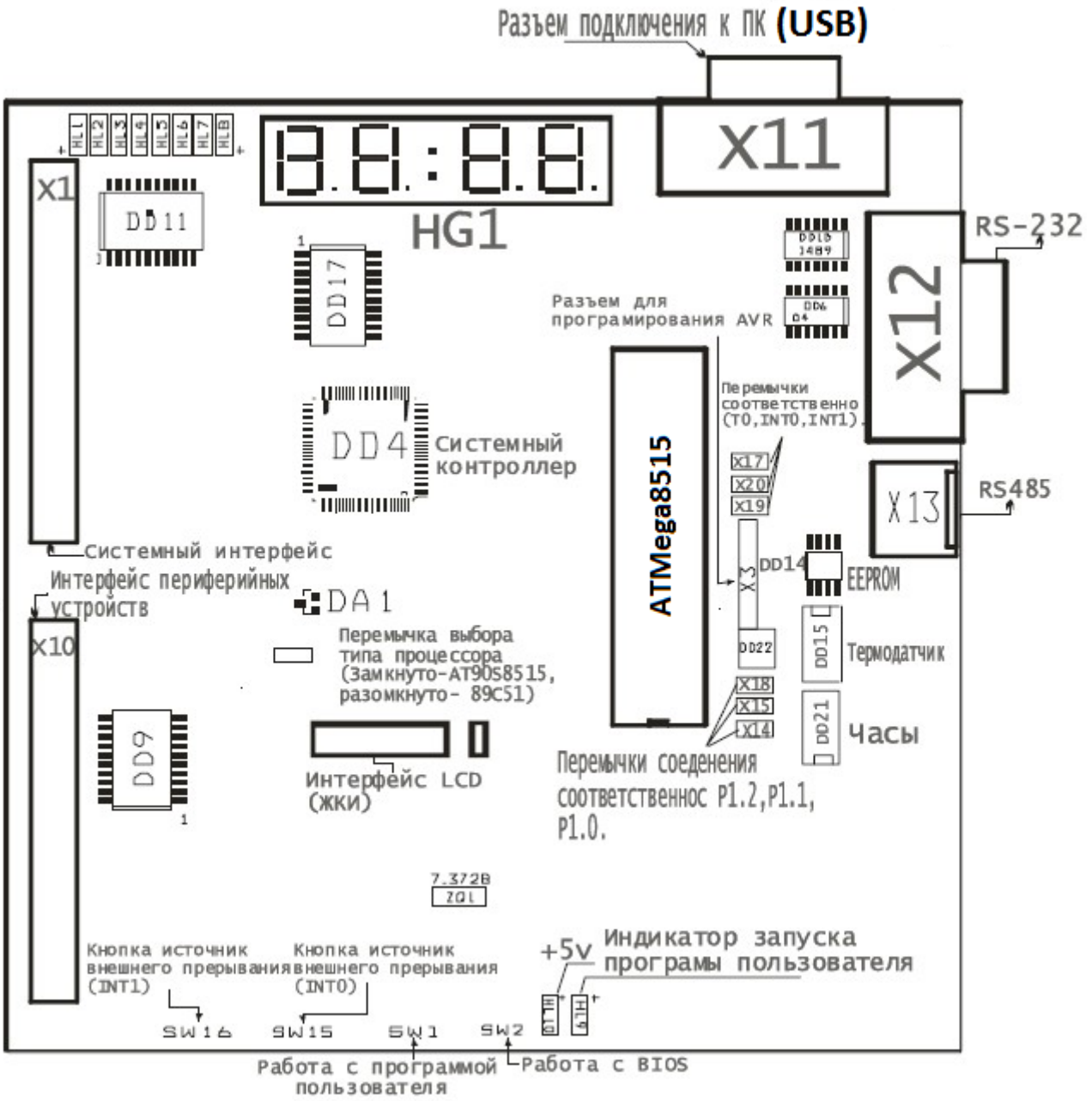
— *k* — константа (размер зависит от инструкции), может быть константное выражение

— *q* — константа (6 бит), может быть константное выражение

— *Rd*l — *R*24, *R*26, *R*28, *R*30 — для инструкций *ADIW* и *SBIW*

— *X,Y,Z* — регистры косвенной адресации (*X*=*R*27:*R*26, *Y*=*R*29:*R*28, *Z*=*R*31:*R*30)

ПРИЛОЖЕНИЕ Б
Схема расположения элементов основного модуля отладочного стенда



*X*1 — Системный интерфейс с полным адресным пространством;

*X*3 — Интерфейс для программирования отладочного стенда;

*X*10 — Интерфейс периферийных устройств;

*X*11 — Интерфейс *USB* для связи стенда с *PC*;

*X*12 — Интерфейс последовательного порта для связи стенда с другими устройствами, имеющими стандартный порт *RS*232*C*;

*X*14, *X*15 — Перемычки подключения устройств с шиной I^2C к микроконтроллеру;

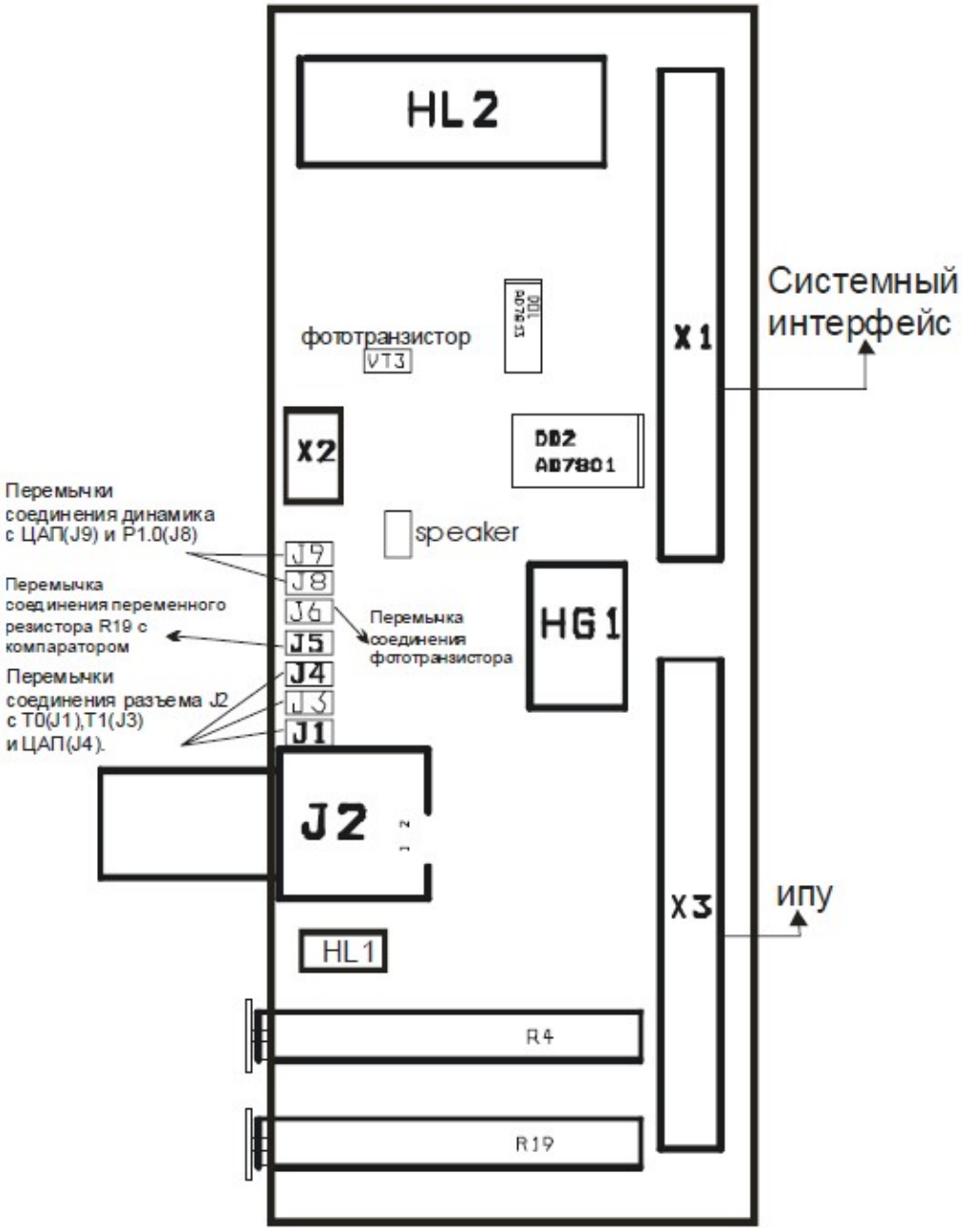
*SW*1 — Кнопка, при нажатии которой происходит перезапуск программы, уже загруженной в память программ стенда;

*SW*2 — Кнопка, нажатие которой разрешает повторную загрузку программы из *PC* в память стенда (при нажатии зажигается светодиод *HL*9);

*SW*15 — кнопка, при нажатии которой формируется сигнал запроса прерывания на входе *INT*0 микроконтроллера;

*SW*16 — кнопка, при нажатии которой формируется сигнал запроса прерывания на входе *INT*1 микроконтроллера.

ПРИЛОЖЕНИЕ В
Схема расположения элементов модуля расширения отладочного стенда



*HG*1 — знакосинтезирующий (точечный) индикатор 7×5;

*HL*1 — светодиодный индикатор срабатывания компаратора;

*HL*2 — четырехразрядный семисегментный динамический индикатор;

*R*4 — переменный резистор, предназначенный для регулировки входного сигнала АЦП;

*R*19 — переменный резистор, предназначенный для изменения частоты генератора с управляемой частотой;

Заказ № _____ от “ ____ ” _____ 2011 г. Тираж ____ экз.
Изд-во СевНТУ