**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение
высшего образования Российской Федерации
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

**В.А. БОЛЬШАКОВ, В.А. МИКЛУШ**

# МИКРОКОНТРОЛЛЕРЫ

## Лабораторный практикум

РГГМУ

Санкт-Петербург
2017

УДК 621.391.23

Составители: Большаков В.А., канд. тех. наук, доцент, Миклуш В.А.

Рецензент: В.В. Соснин, канд. тех. наук, доцент кафедры вычислительной техники университета ИТМО

В практикум включены лабораторные работы, выполняемые студентами гидрометеорологического университета, обучающимися по специальности “Морские информационные системы и оборудование”. Практикум предназначен для ознакомления с принципами построения микропроцессорных систем на основе микроконтроллеров и обучения основным средствам и методам программирования микроконтроллеров. Курс включает в себя 12 лабораторных работ, в которых использованы интегрированная отладочная среда AVR Studio фирмы Atmel, лабораторные макеты на основе однокристальной микро-ЭВМ 87C51FA и оценочные комплекты разработчика C8051F064EK. Каждая лабораторная работа содержит описание необходимых для ее выполнения технических средств, задания и вопросы для самопроверки знаний. В описаниях лабораторных работ 5 – 11 использованы материалы методических указаний [3], разработанных для них поставщиком лабораторных макетов.
Полученные с помощью лабораторного практикума знания и навыки работы с программно-аппаратными средствами микроконтроллеров типа MCS51 и AVR, могут использоваться при курсовом и дипломном проектировании.

## Введение

Микроконтроллеры стали в наше время одним из основных элементов электронной аппаратуры. Они применяются практически во всех технических устройствах и системах. Поэтому специалистам, работающим с современной электроникой, необходимо знать основы аппаратно-программной организации микроконтроллеров и микропроцессорных систем на их основе.

Выбор в качестве лабораторной базы микроконтроллеров семейств AVR и MCS51 обусловлен их широким распространением. Микроконтроллеры семейства MCS51 [1,2,6], начало которому положила фирма Intel еще в 1980 г. стали классикой и выпускаются многими фирмами. Микроконтроллеры семейства AVR фирмы Atmel [4,5] приобрели большую популярность благодаря выгодному соотношению цена/быстродействие/энергопотребление, удобным режимам программирования, доступности программно-аппаратных средств поддержки разработок и широкой номенклатуре. Микроконтроллеры семейств MCS51 и AVR многоцелевого назначения представляют собой удобный инструмент для создания современных высокопроизводительных встраиваемых в аппаратуру управляющих микропроцессорных систем.

Лабораторный практикум предназначен для обучения программированию микроконтроллеров при решении целевых задач, связанных с проектированием микропроцессорных систем различного назначения. В лабораторных работах экспериментально исследуются устройства и режимы функционирования микроконтроллеров, что позволяет студентам развить основные навыки создания целевых проектов.

Лабораторный цикл включает в себя разработку программ и экспериментальное исследование принципов управления аппаратными средствами микроконтроллеров AVR фирмы Atmel и микроконтроллеров с ядром MCS51.

Для лабораторных исследований микроконтроллеров AVR фирмы Atmel используются симулятор интегрированной отладочной среды AVR Studio. Микроконтроллеры семейства MCS51 изучаются в лабораторных работах на макетах с однокристальной микро-ЭВМ 87C51FA и оценочных комплектах C8051F064EK [7].

Эти средства позволяют студентам вместе с изучением и исследованием микроконтроллеров получить практические навыки разработки микропроцессорных систем.

## Порядок выполнения и оформления лабораторных работ

Выполнение каждой лабораторной работы состоит из нескольких этапов и должно начинаться с изучения задания и ознакомления с теоретическими сведениями, приведенными в описании работы. При выполнении каждого задания целесообразно составить алгоритм, включающий последовательность действий, необходимых для реализации проекта.

Отчеты по лабораторным работам представляются в электронном виде. Требования к содержанию отчетов указаны в текстах лабораторных работ. Отчет должен быть оформлен с соблюдением ГОСТа и иметь титульную страницу. В верхней части титульной страницы указывается название университета и кафедры. Затем название и номер лабораторной работы, номер группы и фамилии выполнивших ее студентов. Пример оформления титульной страницы приведён в приложении 7. Материалы отчетов могут также распечатываться для их защиты при наличии принтера и расходных материалов к нему. В конце отчета следует привести перечень литературы, использованной при подготовке к лабораторной работе, ее выполнении и оформлении отчета.

# Технические средства микроконтроллера AT90S8515, симулятор ASTUDIO и система команд

## Устройства памяти

Микроконтроллер AT90S8515 имеет следующие ресурсы памяти:

- 8 Кбайт программной памяти FLASH;
- 512 байт энергонезависимой памяти EEPROM;
- 512 байт внутренней оперативной памяти SRAM;
- 64 регистра ввода/вывода;
- 32 регистра общего назначения.

## Регистр слова состояния - SREG

Регистр **SREG** – размещен в пространстве I/O по адресу $3F(5F)/

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $3(F)5F | **I** | **T** | **H** | **S** | **V** | **N** | **Z** | **C** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – I Разрешение глобального прерывания**

Бит разрешения глобального прерывания для разрешения прерывания должен быть установлен в состояние 1. Управление разрешением конкретного прерывания выполняется битом маски прерывания в соответствующем регистре. Если бит глобального прерывания очищен (в состоянии 0), то ни одно из разрешений прерываний, установленных в этих регистрах, не действует. Бит I аппаратно очищается после прерывания и автоматически устанавливается опять после выполнения подпрограммы обработки прерывания командой RETI.

**Bit 6 – T Бит сохранения копии**

Команды копирования бита BLD (Bit LoaD) и BST (Bit STore) используют бит T как бит источник и бит назначения при операциях с битами. Заданный разряд любого РОН может быть скопирован в этот разряд регистра SREG командой BST или установлен в соответствии с содержимым данного разряда командой BLD.

**Bit 5 – H – Флаг полупереноса**

Флаг полупереноса указывает на перенос из младшей тетрады байта в старшую или на заем из старшей тетрады при выполнении арифметических операций.

**Bit 4 – S – S=N ⊕ V – Бит знака**

Бит S находится в состоянии, определяемом логической операцией исключающего ИЛИ между флагом отрицательного значения N и флагом V переполнения числа в дополнительном коде. Устанавливается в «1», если результат выполнения арифметической операции меньше нуля.

**Bit 3 – V – Флаг переполнения дополнительного кода**

Флаг V устанавливается в «1» при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами, представленными в дополнительном коде.

**Bit 2 – N – Флаг отрицательного значения**

Этот флаг устанавливается в «1», если старший (7-й) разряд результата операции равен «1».

**Bit 1 Z – Флаг нулевого значения**

Бит Z устанавливается в «1», если результат выполнения операции равен нулю.

**Bit 0 C– Флаг переноса**

Флаг переноса устанавливается в «1», если в результате выполнения операции произошел выход за границы байта.

# Симулятор интегрированной программной среды ASTUDIO

**Ассемблер**

Наиболее эффективные программы обычно получаются при использовании языка Ассемблер, хотя при этом также увеличивается сложность и время разработки программы. Для работы с микроконтроллером созданы интегрированные отладочные средства, которые включают в себя компилятор с языка ассемблера и позволяют полностью контролировать выполнение программ с использованием симулятора, поддерживающего различные типы микроконтроллеров.

Для микроконтроллеров семейства AVR имеется бесплатно распространяемая фирмой Atmel интегрированная среда AVRStudio, которая поддерживает в режиме симуляции выполнение программ, написанных на ассемблере в виде текста формата AVR Assembler, IAR Systems Assembler и в формате языка С компилятора фирмы IAR Systems ICCASO C Compiler.

Пользователь может выполнять программу полностью в пошаговом режиме, трассируя блоки функций, или выполняя программу до места, где стоит курсор. В дополнение можно определять неограниченное число точек останова, каждая из которых может быть включена или выключена.

**Создание проекта в AVR Studio**

Для работы над новым проектом в среде AVR Studio 3.10 создается папка проекта с произвольным именем. В папку необходимо скопировать файл инициализации выбранного для реализации проекта микроконтроллера (в нашем случае **“8515def.inc”**). Этот файл входит в состав прикладного программного обеспечения AVRStudio. Далее производится запуск программы AVR Studio.

В результате запуска на экране появляется окно запуска программы.

Для создания нового проекта необходимо выбрать в меню **Project\New**. В появившемся диалоговом окне, нужно ввести имя проекта (**Project name**) и его расположение (**Location**) и выбрать тип проекта (**AVR Assembler**, если используется язык ассемблер и **Generic 3d party C compiler**, если используется язык С).

После выбора типа проекта (**AVR Assembler)** и нажатия кнопки **ОК** на экране появляется окно организатора проекта, показывающее все связанные с проектом файлы.

Щёлкнуть правой кнопкой мыши на **папке Assembler Files** и, выбрав **Add Fil** добавить в неё файл программы на языке ассемблер. Это может быть сделано двумя способами: в проект добавляется уже существующий файл с расширением **.asm** или создается новый. Таким же образом необходимо в папку **Other File** поместить файл инициализации **“8515def.inc”.**

В более современной среде AVR Studio 4.19 подключать “8515def.inc” не требуется.

**Ассемблирование программы и исправление ошибок**

После создания проекта осуществляются ассемблирование, исправление ошибок и отладка программы. Для ассемблирования (трансляции программы) в папке **Project** выбирается пункт **Assemble.** В открывающемся после ассемблирования окне **Project Output** содержатся сообщения ассемблера о количестве слов программы, наличии ошибок и месте их нахождения. В сообщении об ошибке указываются номер сроки, в которой она обнаружена и краткое текстовое сообщение о характере ошибки.

Двойной щелчок на сообщении об ошибке в окне сообщений приводит к тому, что окно текстового редактора исходного текста становится активным, а курсор становится в начало строки с ошибкой. Таким образом исправляются все ошибки. Кроме того, следует учесть, что существуют взаимозависимые ошибки, т.е. ошибки, которые следуют одна из другой. Поэтому рекомендуется ассемблировать файл по новой после исправления каждой ошибки, для выявления и исправления всех независимых друг от друга ошибок. После успешного исправления всех ошибок в окне сообщения об ошибках появляется сообщение об успешном завершении трансляции.

**Запуск и отладка программы**

Когда все ошибки будут исправлены, программу можно запустить нажатием кнопки **RUN.** При этом открывается диалоговое окно выбора опций симулятора **Simulator Options**, где выбираются нужный микроконтроллер (AT90S8515) и тактовая частота процессорного ядра

(Frequency). После этого программа выполняется. Выполнение программы останавливается нажатием кнопки **Break.**

Результаты работы программы можно проанализировать в раскрывающихся диалоговые окнах:

- **Registers** – диалоговое окно состояния регистров общего назначения;
- **IO** – диалоговое окно устройств ввода/вывода;
- **Memory** – диалоговое окно состояния памяти микроконтроллера.

Далее осуществляется отладка программы.

На рисунке 1 представлен общий вид окна симулятора **AVRStudio.**

**Система команд микроконтроллера AT90S8515**

Система команд приведена в приложении 1.

Команды микроконтроллеров семейства AVR делятся на:

- арифметические и логические;
- передачи управления;
- пересылки данных;
- сдвига и операций с битами с битами.

Транслятор ассемблера не различает строчные и прописные буквы. Однако для удобства понимания программы иногда рекомендуется команды и метки записывать строчными (маленькими) буквами, а директивы прописными (большими).

В командах ассемблера могут использоваться выражения, содержащие помимо операндов и операций функции:

LOW(выражение) – возвращает младший байт выражения;

HIGH(выражение) – возвращает старший байт выражения;

BYTE2(выражение) – возвращает 2 байта выражения;

BYTE3(выражение) – возвращает 3 байта выражения;

BYTE4(выражение) – возвращает 4 байта выражения;

LWRD(выражение) – возвращает биты 0 – 15 выражения;

HWRD(выражение) – возвращает биты 16– 31 выражения.

Перечень и описания команд содержатся в приложении 1 практикума.

Приложение 2 практикума содержит список регистров ввода/вывода микроконтроллера AT90S8515.

**Директивы ассемблера**

Директивами называются указания транслятору о необходимости выполнения определяемых ими действий в процессе трансляции программы. При этом сами директивы не транслируются и не входят в состав исполняемого микроконтроллером модуля. Перечень и описание основных директив ассемблера приведены в приложении 3 практикума.

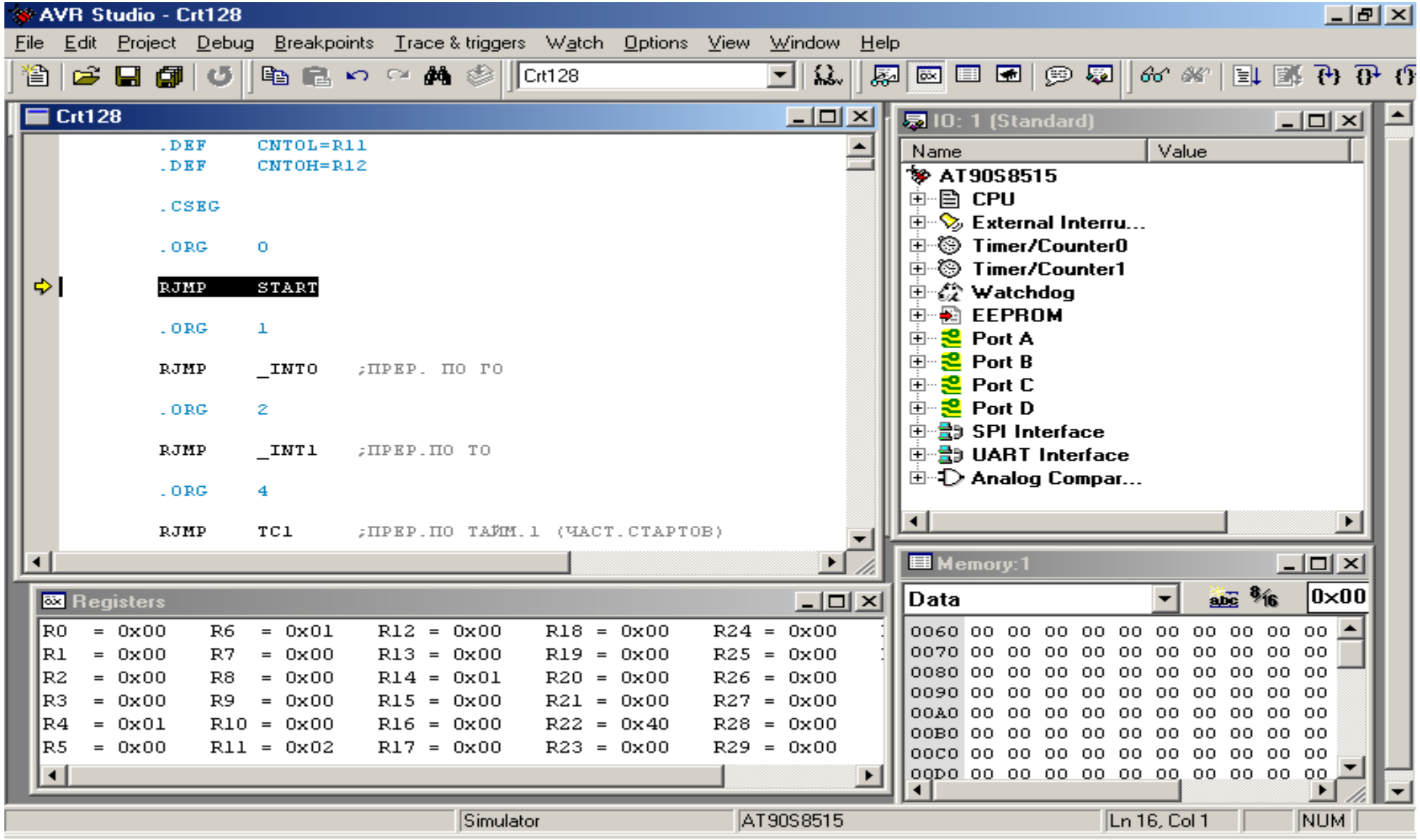



Рис. 1. Общий вид окна симулятора AVR Studio

# Лабораторная работа № 1
# Программирование памяти и системы прерываний микроконтроллера AT90S8515

**Цель работы:** научиться создавать файлы в среде Astudio, определять сегменты программы, инициализировать прерывания и стек и проверять работу программы.

## Теоретические сведения

Коды программ микроконтроллера AT90S8515 записываются в 8 Кбайт встроенной памяти. На кристалле также размещены 512 байт внутренней оперативной памяти SRAM и 512 байт энергонезависимой памяти EEPROM;

**Конфигурация памяти.**

На рис. 2 представлена конфигурация памяти микроконтроллера AT90S8515.

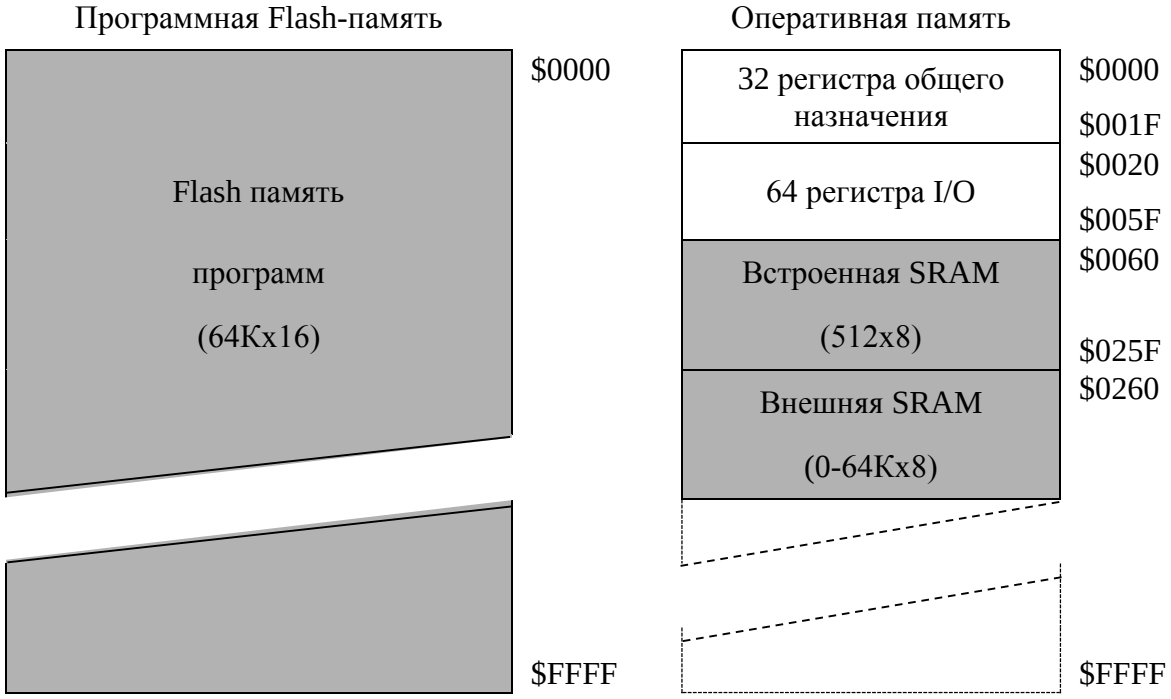



Рис. 2. Конфигурация памяти.

### Управление прерываниями

Микроконтроллер AT90S8515 поддерживает 12 различных источников прерываний. Каждое из этих прерываний и сброс имеют разные адреса векторов в области программной памяти. Все прерывания назначаются индивидуальными битами разрешения, которые должны быть установлены вместе с битом I регистра слова состояния SREG.

За векторами сброса и прерывания закреплены младшие адреса памяти программ. Полный список этих адресов приведен в таблице. Меньшие значения адреса соответствуют более высокому уровню приоритета. Сброс имеет наивысший приоритет.

Таблица. 1.1 Адреса векторов сброса и прерываний.

| № вектора | Программный адрес | Обозначение | Определение прерываний |
|---|---|---|---|
| 1 | $0000 | RESET | Внешнее прерывание, сброс по включению питания и сторожевому таймеру |
| 2 | $0001 | INT0 | Запрос внешнего прерывания 0 |
| 3 | $0002 | INT1 | Запрос внешнего прерывания 1 |
| 4 | $0003 | TIMER1 CAPT | Захват таймера/счетчика 1 |
| 5 | $0004 | TIMER1 COMPA | Совпадение А при сравнении таймера/счетчика 1 |
| 6 | $0005 | TIMER1 COMPB | Совпадение В при сравнении таймера/счетчика 1 |
| 7 | $0006 | TIMER1 OVF | Переполнение таймера/счетчика 1 |
| 8 | $0007 | TIMER0 OVF | Переполнение таймера/счетчика 0 |
| 9 | $0008 | SPI, STC | Завершение пересылки по интерфейсу SPI |
| 10 | $0009 | UART, RX | UART, завершение приема |
| 11 | $000A | UART, UDRE | UART, регистр данных пуст |
| 12 | $000B | UART, TX | UART, передача завершена |
| 13 | $000C | ANA_COMP | Аналоговый компаратор |

**ПРИМЕР 1** – начало программы, определение сегментов памяти программы, ОЗУ и EEPROM.

```
.DEVICE AT90S8515                ;тип микросхемы
.INCLUDE "8515def.inc"           ;подключение к основному файлу файла
                                 ;инициализации 8515def.inc, который является
                                 ;частью документации к контроллеру
.CSEG                            ;директива определения сегмента программного кода
.ORG  0x00                       ;директива указания начального адреса сегмента
 rjmp start                       ;сброс микроконтроллера и переход на метку
 start:                           ;start начала программы
.DSEG                            ;определение сегмента данных (ОЗУ)
.ORG  0x60                       ;указание начального адреса сегмента
.ESEG                            ;определение сегмента EEPROM
.ORG  0x00                       ;указание начального адреса EEPROM
```

**ПРИМЕР 2 –** определение векторов прерываний и стека

```
.CSEG                    ;директива определения сегмента программного кода
.ORG  0x00               ;директива указания начального адреса сегмента
rjmp start               ;сброс микроконтроллера и переход на метку
                         ;start начала программы
ORG   0x01               ;директива указания адреса 0x01 вектора внешних
                         ;прерываний по входу  INT0
rjmp in0                 ;переход к подпрограмме обработки прерывания
```

```
in0:                        ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
.ORG  0x02                  ;директива указания адреса 0x02 вектора внешних
                            ;прерываний по входу  INT1
rjmp in1                    ;переход к подпрограмме обработки прерывания
in1:                        ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
.ORG  0x04                  ;директива указания адреса 0x04 вектора
                            ;прерываний по выходу сравнения A таймера/счётчика 1
rjmp tc1coma                ;переход к подпрограмме обработки прерывания
tc1coma:                    ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
.ORG  0x05                  ;директива указания адреса 0x05 вектора
                            ;прерываний по выходу сравнения B таймера/счётчика 1
rjmp tc1comb                ;переход к подпрограмме обработки прерывания
tc1comb:                    ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
.ORG  0x07                  ;директива указания адреса 0x07 вектора
                            ;прерываний по выходу переполнения таймера/счётчика 0
rjmp tc0ovf                 ;переход к подпрограмме обработки прерывания
tc0ovf:                     ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
.ORG  0x09                  ;директива указания адреса 0x09 вектора
                            ;прерываний по завершению приёма интерфейса  UART
rjmp rxrdy                  ;переход к подпрограмме обработки прерывания
rxrdy:                      ;метка начала подпрограммы
reti                        ;выход из подпрограммы обработки прерывания
start:
ldi     R16,HIGH(RAMEND) ;инициализация стека, начальный адрес вершины стека
out     SPH,R16           ;обычно задаётся равным максимальному адресу ОЗУ,
ldi     R16,LOW(RAMEND) ; так как стек растёт в сторону младших адресов оперативной
out     SPL,R16           ;памяти
m: rjmp m                 ;бесконечный цикл
```

**ПРИМЕР 3** – резервирование памяти в сегменте  ОЗУ

```
        .DSEG                   ;определение сегмента данных (ОЗУ)
        .ORG  0x60              ;указание начального адреса сегмента
fl0:        .BYTE 1                 ;резервируется 1 байт под FL0
fl1:        .BYTE 1                 ;резервируется 1 байт под FL1
cnto:       .BYTE 2                 ;резервируются 2 байта под CNTO
atin:       .BYTE 2                 ;резервируются 2 байта под ATIN
cntb:       .BYTE 1                 ;резервируется 1 байт под CNTB
cntb1:  .    BYTE 1                 ;резервируется 1 байт под CNTB1
bufirs:     .BYTE 24                ;резервируются 24 байта под BUFIRS
bufors:     .BYTE 24                ;резервируются 24 байта под BUFORS
nugt:       .BYTE 98                ;резервируются 98 байт под NUGT
regsr:      .BYTE 98                ;резервируются 98 байт под REGSR
chis1:      .BYTE 1                 ;резервируется 1 байт под ЕДИНИЦЫ
chis2:      .BYTE 1                 ;резервируется 1 байт под ДЕСЯТКИ
chis3:      .BYTE 1                 ;резервируется 1 байт под СОТНИ
```

```
chis4:      .BYTE 1                 ;резервируется 1 байт под ТЫСЯЧИ
hexf:       .BYTE 2                 ;резервируются 2 байта под HEXF
ends:       .BYTE 1                 ;резервируется 1 байт (конец сегмента ОЗУ)
```

**ПРИМЕР 4** – заполнение сегментов EEPROM

```
        .ESEG                       ;организация EEPROM
        .ORG  0x00
esint:  .DB   $F8,$24,$00,$00,$00   ;запись в EEPROM 5-ти констант в
                                    ;шестнадцатиричном коде
edsint: .DB   $7A,$E2,$00           ;запись в EEPROM 3-х констант в
                                    ;шестнадцатиричном коде
etcont: .BYTE 2                     ;резервируются 2 байта под ETCONT
rfmet:  .BYTE 2                     ;резервируются 2 байта под RFMET
ramet:  .BYTE 2                     ;резервируются 2 байта под RAMET
riket1: .BYTE 2                     ;резервируются 2 байта под RIKET1
riket2: .BYTE 2                     ;резервируются 2 байта под RIKET2
etalon: .BYTE 256                   ;резервируются 256 байт под ETALON
```

## Задание

1. Создать файл в среде ASTUDIO.
2. Определить в качестве начальной части программы стандартные директивы .DEVICE AT90S8515 и .INCLUDE "8515def.inc".
3. Определить сегменты программного кода, ОЗУ и EEPROM.
4. Заполнить сегмент памяти программного кода в соответствии с вариантом, заданным в Табл. 1.2, а сегменты ОЗУ и EEPROM по указанию преподавателя.
5. Проверить работу программы в симуляторе ASTUDIO.

Таблица 1.2. Варианты заданий

| Вариант | RESET | INT0 | INT1 | TIMER1 COMPA | TIMER1 COMPB | TIMER1 OVF | TIMER0 OVF | UART, RX | UART, UDRE | UART, TX | ANA_ COMP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1. | + | + | | + | | | + | + | + | | + |
| 2. | + | | + | | + | | + | + | | + | + |
| 3. | + | + | | | | + | + | + | + | | + |
| 4. | + | | + | + | | | + | + | | + | + |
| 5. | + | + | | | + | | + | + | + | | + |
| 6. | + | | + | | | + | + | + | | + | + |
| 7. | + | + | | + | | | + | + | + | | + |
| 8. | + | | + | | + | | + | + | | + | + |
| 9. | + | + | | | | + | + | + | + | | + |
| 10. | + | | + | + | | | + | + | | + | + |

Знак + в таблице указывает, какие векторы прерываний нужно инициализировать.

Контрольные вопросы для самопроверки.

1. Какие ресурсы памяти имеет микроконтроллер AT90S8515?
2. Какие биты содержит регистр состояния SREG?
3. Конфигурация оперативной памяти микроконтроллера AT90S8515?
4. Что такое прерывание и как работает микроконтроллер в этом режиме?
5. Векторы сброса и прерываний. Чем определяются уровни приоритета векторов?
6. Для чего нужны директивы DEVICE и INCLUDE?
7. Какие сегменты памяти есть в микроконтроллере?

8. Как организованы память программ и EEPROM?
9. Какими директивами записываются константы в память программ и EEPROM?
10. Как резервируется память?

## Лабораторная работа № 2
## Организация работы параллельных портов и последовательного интерфейса UART.

**Цель работы:** научиться программировать работу параллельных портов микроконтроллера и последовательного интерфейса UART.

### Теоретические сведения

Контроллер AT90S8515 имеет четыре 8-ми разрядных порта ввода-вывода – порты A, B, C, D. Каждый порт имеет три регистра для организации его работы:

- регистр направления данных;
- регистр данных;
- выводы порта.

Все линии порта могут программироваться независимо друг от друга на ввод или вывод данных Рассмотрим работу с этими регистрами на примере порта A.

**PORTA** – регистр данных порта A, адрес $1B($3B).

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $1B($3B) | **PORTA7** | **PORTA6** | **PORTA5** | **PORTA4** | **PORTA3** | **PORTA2** | **PORTA1** | **PORTA0** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Если бит PORTAn установить в 1, то к выводу n подключается нагрузочный (подтягивающий) MOS резистор и линия работает как входная при программировании её на ввод данных. При программировании как выходной на неё выводится состояние этого бита.

**DDRA –** регистр, определяющий направление передачи данных (ввод или вывод), адрес $1A($3A).

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $1A($3A) | **DDA7** | **DDA6** | **DDA5** | **DDA4** | **DDA3** | **DDA2** | **DDA1** | **DDA0** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Биты DDAn регистра DDRA определяют направление работы соответствующего вывода. При установленном в состояние 1 бите DDAn вывод PAn конфигурируется как вывод выхода. При очищенном бите DDAn (сброшенном в 0) вывод PAn конфигурируется как вывод входа.

**PINA** – не является регистром порта A в полном смысле этого слова. По его адресу осуществляется доступ к значениям на каждого из физических выводов порта. Этот регистр предназначен только для чтения состояний на выводах микросхемы. При чтении PORTA читаются данные из регистра-защелки данных, при чтении PINA – читаются логические значения, соответствующие фактическому состоянию выводов порта.

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $19($39) | **PINA7** | **PINA6** | **PINA5** | **PINA4** | **PINA3** | **PINA2** | **PINA1** | **PINA0** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Работа порта А в качестве цифрового устройства ввода/вывода**

Таблица 2.1. Воздействие битов DDRAn и PORTAn на режим работы выводов порта А.

| DDRAn | PORTAn | I/O | Нагрузочный резистор | Описание |
|---|---|---|---|---|
| 0 | 0 | Вход | Не подключен | Третье состояние (Hi-Z) |
| 0 | 1 | Вход | Подключен | При низком уровне PAn обеспечивают вытекающий ток |
| 1 | 0 | Выход | Не подключен | Низкий уровень, двухтактный выход |
| 1 | 1 | Выход | Не подключен | Высокий уровень, двухтактный выход |

**Примечание: PAn – выводы I/O общего назначения, n=7,6, … 1, 0 – номера выводов порта А.**

**ПРИМЕР 1** Пины 0-3 порта А ориентированы на прием, пины 4-7 – на передачу, регистр данных обнулен, т.е. по линиям 4-7 передаются значения 0, а линии 0-3 находятся в состоянии Z/

```
ldi     R16,$F0
out     DDRA,R16
ldi     R16,$00
out     PORTA,R16
```

**ПРИМЕР 2** Установка 3-го разряда регистра данных в 1 и обнуление 0-го разряда регистра данных

```
sbi    PORTA,3
cbi    PORTA,0
```

**ПРИМЕР 3** Чтение состояния выводов порта А в регистр R16 и выдача данных из регистра R16 в регистр данных порта А

```
in   R16,PINA
out  PORTA,R16
```

Все порты имеют альтернативные функции выводов. Для PORTB и PORTD они приведены в приложении 5 практикума.

При использовании альтернативных функций выводов регистр направления данных и регистр данных должны быть сконфигурированы на ввод/вывод в соответствии с описанием альтернативных функций.

**Служебные регистры MCUCR, GIMSK и GIFR**

**MCUCR** – регистр управления микроконтроллера (MCU CONTROL REGISTER).

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $35 ($55) | **SRE** | **SRW** | **SE** | **SM** | **ISC11** | **ISC10** | **ISC01** | **ISC00** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТ 7 – SRE**: бит подключения внешней памяти (External Sram Enable):
Когда этот бит в состоянии лог. 1 внешняя оперативная память подключена и альтернативные функции: адрес памяти – линии A0 – 7 (PORT A), A8 – 15 (PORTC) и флаги приёма и передачи WR и RD (PORT D) активированы.
**БИТ 6 – SRW**: бит подключения состояния ожидания доступа к внешней памяти:
Когда этот бит в состоянии лог. 1 микроконтроллер 1 цикл ждет доступа к внешней памяти.
**БИТ 5 – SE**: бит разрешения работы микроконтроллера со SLEEP –инструкцией работы в режиме пониженного энергопотребления (спящем).
**БИТ 4 – SM**: установки микроконтроллера в режим SLEEP пониженного энергопотребления.
**БИТЫ 3, 2, 1, 0** – ISC11, ISC10, ISC01, ISC00: управляющие биты, определяющие режимы работы по входам внешних прерываний INT1 и INT0.
Состояния этих битов и соответствующие им режимы прерываний приведены в Табл. 2.2.

Таблица 2.2. Режимы прерываний (x =0 или 1 номер входа прерывания)

| ICSx1 | ICSx0 | Описание |
|---|---|---|
| 0 | 0 | Прерывание генерируется уровнем лог. 0 |
| 0 | 1 | Резерв |
| 1 | 0 | Прерывание генерируется отрицател. фронтом |
| 1 | 1 | Прерывание генерируется положител. фронтом |

**GIMSK** – Общий регистр маски прерываний (GENERAL INTERRUPT MASK REGISTER)

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $3B ($5B) | **INT1** | **INT0** | - | - | - | - | - | - |
| Чтение/Запись | R/W | R/W | R | R | R | R | R | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТ 7 – INT1*:*** запрос внешнего прерывания 1 разрешен. Когда этот бит установлен, а также установлен бит **I**, в регистре SREG, разрешается прерывание от внешнего вывода. Биты управления запуском прерывания (ISC11 и ISC10) в MCUCR определяют, по какому событию обрабатывается прерывание. При возникновении прерывания выполняется переход по адресу $002 для выполнения подпрограммы обработки прерывания.
**БИТ 6 – INT0:** запрос внешнего прерывания 0 разрешен. Когда этот бит установлен, а также установлен бит **I**, в регистре SREG, разрешается прерывание от внешнего вывода. Биты управления запуском прерывания (ISC01 и ISC00) в MCUCR определяют, по какому событию обрабатывается прерывание. При возникновении прерывания выполняется переход по адресу $001 для выполнения подпрограммы обработки прерывания.
**БИТЫ 5…0** – в AT90S8515 э зарезервированы и всегда читаются как 0.

**GIFR** – Общий регистр флагов прерываний (**GENERAL INTERRUPT FLAG REGISTER**)

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $3A ($5A) | **INTF1** | **INTF0** | - | - | - | - | - | - |
| Чтение/Запись | R/W | R/W | R | R | R | R | R | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***БИТ 7 - INTF1:*** флаг внешнего прерывания 1. При возникновении на выводе INT1 события, вызывающего прерывание, INTF1 устанавливается в 1. Если установлены бит I в регистре SREG и бит INT1 в GIMSK, происходит переход на вектор прерывания по адресу $002. Флаг очищается после обработки прерывания. Кроме того, флаг может быть очищен при записи в него логической «1».

***БИТ 6 – INTF0:*** флаг внешнего прерывания 0. При возникновении на выводе INT0 события, вызывающего прерывание, INTF0 устанавливается в 1. Если установлены бит I в регистре SREG и бит INT0 в GIMSK, происходит переход на вектор прерывания по адресу $001. Флаг очищается после обработки прерывания. Кроме того, флаг может быть очищен при записи в него логической «1».

***БИТЫ 5…0*** – в AT90S8515 зарезервированы и всегда читаются как 0.

**UART - универсальный асинхронный приемопередатчик**

**UART** – универсальный асинхронный последовательный приемопередатчик (Universal Asynchronous Receiver and Transmitter), является стандартным функциональным узлом микропроцессоров AVR. Основные характеристики UART:

- возможность разных значений скорости обмена информацией;
- 8 или 9 битов данных;
- мажоритарная фильтрация шума;
- определение переполнения;
- детектирование ошибки кадра;
- определение неверного стартового бита;
- три раздельных прерывания (завершение передачи, очистка регистра передачи и завершение приема).

В контроллере AT90S8515 UART обладает следующими возможностями:

- на частотах работы контроллера от1 MHz до 11.059 MHz позволяет осуществлять передачу данных со скоростями 2400, 4800, 9600, 14400, 19200, 28800, 57600, 115200(BAUD), а на частотах от 14.746 MHz до 20 MHz со скоростями 2400, 4800, 9600, 14400, 19200, 28800, 57600 (BAUD);
- осуществляет прием и передачу 8-ми или 9-ти битных данных;
- имеет три вектора прерываний по адресам:

  1) $009 - UART, Rx Complete (прерывание по окончании приема данного);

  2) $00A – UART, Data Register Empty (прерывание по заполнению регистра данных UART);

  3) $00B – UART, Tx Complete (прерывание по окончании передачи данных).

Управление и организация работы UART осуществляется через следующие регистры:

- **UDR** (UART I/O DATA REGISTER)- 8- ми разрядный регистр данных по адресу $0C($2C), доступен для чтения и записи;
- **UBRR** (BAUD RATE REGISTER)- 8-ми разрядный регистр, определяющий BAUD скорости передачи данных, доступен для чтения и записи;
- **UCR** (UART CONTROL REGISTER)- 8-ми разрядный регистр управления: биты 2-7 доступны для чтения и записи, 1–только для чтения, 0-только для записи;
- **USR** (UART STATUS REGISTER)- 8-ми разрядный регистр состояния UART, доступный только для чтения.

**USR –** РЕГИСТР СОСТОЯНИЯ UART:

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $0B ($2B) | RXC | TXC | UDRE | FE | OR | - | - | - |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТ 7 – RXC:** флаг окончания приема (UART Receiver Complete).
Этот бит устанавливается, когда принятые данные переместились из сдвигающего регистра в UDR и очищается, когда данные изъяты из UDR.
**БИТ 6 – TXC:** флаг окончания передачи (UART Transmitter Complete).
Этот бит устанавливается, когда посылка, включая стоп-бит, полностью выдвинута из передающего сдвигающего регистра и UDR готов принять новые данные. TXC очищается, если бит TXCIE в UCR установлен в лог. 1.
**БИТ 5 – UDRE:** флаг готовности регистра данных (UART Data Register Empty).
Этот бит устанавливается, когда записанная в UDR посылка данных переместилась в передающий сдвигающий регистр. Установка этого флага показывает, что передатчик готов принять в UDR новую посылку данных для её передачи.
**БИТ 4 – FE:** флаг ошибки формата данных (Framing Error).
Этот флаг устанавливается, когда UART не находит стоп-бит в принятой посылке данных, то есть когда стоп-бит – лог. 0, а очищается, когда с приходит стоп-бит, кодированный лог. 1.
**БИТ 3 – OR:** флаг переполнения регистра данных (Over Run).
Этот флаг устанавливается, когда данные при их приёме еще не изъяты из UDR, а в него уже поступает следующая посылка данных.
**БИТ 2-0** – Res: резервные биты.
Резервные биты должны быть в состоянии логического 0.

**UCR –** РЕГИСТР УПРАВЛЕНИЯ UART:

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $0A ($2A) | **RXCIE** | **TXCIE** | **UDRIE** | **RXEN** | **TXEN** | **CHR9** | **RXB8** | **TXB8** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

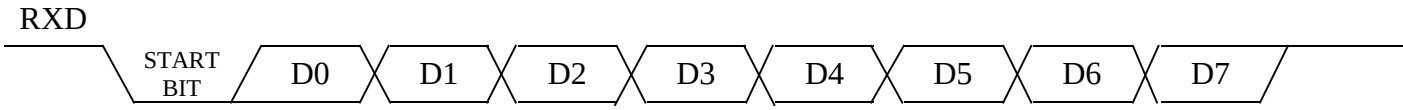
**БИТ 7 – RXCIE:** разрешение прерывания по окончании приема (RX Complete Interrupt Enable).
Когда этот бит в состоянии лог.1 и бит RXC регистра USR установлен в состояние лог. 1, произойдет прерывание микроконтроллера по событию окончания приёма.
**БИТ 6 – TXCIE**: разрешение прерывания по окончании передачи (TX Complete Interrupt Enable):
Когда этот бит в состоянии лог.1 и бит TXC регистра USR установлен в состояние лог. 1, произойдет прерывание микроконтроллера по событию окончания передачи.
**БИТ 5 – UDRIE:** разрешение прерывания по заполнению регистра данных UART (UART Data Register Empty Interrupt Enable):
Когда этот бит в состоянии лог.1 и бит состояния UDRE регистра USR установлен в состояние лог. 1, произойдет прерывание микроконтроллера по событию готовности принять в UDR новую посылку данных для её передачи.
**БИТ 4 – RXEN:** разрешение приема (Receiver Enable):
Когда этот бит в состоянии лог.1, он разрешает UART произвести прием. Если прием не разрешен, флаги состояния RXC, OR и FE в регистре USR не могут установится. Флаги RXC, OR и FE сбрасываются после извлечение данных из регистра UDR.
**БИТ 3 – TXEN:** разрешение передачи (Transmitter Enable):
Когда этот бит в состоянии лог.1, он разрешает UART произвести передачу.
**БИТ 2 – CHR9:** девятибитные числа (9 Bit Characters):
Когда этот бит стоит в состоянии лог.1, принимаемые и передаваемые посылки данных имеют 9-ти битный формат плюс старт- и стоп- биты. 9-й бит часто служит для обнаружения ошибок по признаку чётности (или нечётности) количества единиц в посылке. Он также используется как указатель мультипроцессорного режима работы UART.

**БИТ 1 – RXB8:** разряд для хранения восьмого бита при приёме девятибитных чисел (Receive Data Bit 8):
**БИТ 0 – TXB8:** разряд для хранения восьмого бита при передаче девятибитных чисел (Transmit Data Bit 8):
**UBRR –** РЕГИСТР ОПРЕДЕЛЕНИЯ СКОРОСТИ ПЕРЕДАЧИ ДАННЫХ**.**

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $09 ($29) | **MSB** | - | - | - | - | - | - | **LSB** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**MSB** и **LSB** – старший и младший биты.
В UBRR записывается константа, которая подбирается по таблице “UBRR Setting at Various Crystal Frequencies” приведённой в приложении 4 практикума.

UART имеет два вывода с микросхемы – RXD (приёма) и TXD (передачи), которые являются альтернативными функциями для порта D пины 0-й и 1-й соответственно. Когда прерывания и режимы приема и передачи разрешены соответствующими контрольными битами, эти альтернативные функции задействованы. Их состояние задается при инициализации порта D.
Формат посылки восьмибитного числа




**ПРИМЕР 1** Определение вектора прерывания по событию завершения приёма и переход к подпрограмме обработки запроса прерывания

```
.ORG  9
rjmp  RXRDY
```

**ПРИМЕР 2** Инициализация UART

```
ldi r16,$00              ;установка интерфейса UART на приём данных
out ddrd,r16
ldi r16,$02
out portd,r16
ldi   R16,51         ;установка скорости обмена информацией
out   UBRR,R16
ldi   R16,$90        ;разрешение приёма и прерывания после его завершения
out   UCR,R16
```

**ПРИМЕР 3** Подпрограмма обработки прерывания по завершению приема данных

```
rxrdy:  push  R16        ; сохранение содержимого R16 в стеке
        in    R16,UDR    ;пересылка  принятого байта данных из UDR в R16
        sts   RAM,R16    ; сохранение принятого байта данных в ОЗУ
        pop   R16        ; возврат содержимого R16 из стека
        reti             ; возврат из подпрограммы прерывания
```

## Задание

1. Инициализировать порты A, B и C в соответствии с заданным вариантом

Таблица 2.4. Варианты

| Вариант | A | B | C |
|---|---|---|---|
| 1. | На прием | На передачу | Биты 0-3 – на прием<br>Биты 5-7 – на передачу |
| 2. | На передачу | На прием | Биты 0,2,5,6 – на прием<br>Биты 1,3,4,7 – на передачу |
| 3. | Биты 5-7 – на прием<br>Биты 0-3– на передачу | На передачу | На прием |
| 4. | На прием | Биты1,3,4,7 – на прием<br>Биты 0,2,5,6 – на передачу | На передачу |
| 5. | На передачу | На прием | Биты 0-2 – на прием<br>Биты 3-7– на передачу |
| 6. | Биты 5-7– на прием<br>Биты 0-4– на передачу | На передачу | На прием |
| 7. | На передачу | Биты 1,2,4,5– на прием<br>Биты 0,3,6,7– на передачу | На прием |
| 8. | На передачу | На прием | Биты 2,5,6– на прием<br>Биты 0,1,3,4,7– на пере-дачу |
| 9. | На прием | На передачу | Биты 4,6,7– на прием<br>Биты 0-3,5– на передачу |
| 10. | На передачу | На прием | Биты – на прием<br>Биты – на передачу |

2. Инициализировать порт D с учетом его альтернативных функций для UART.
3. Инициализировать внешние прерывания т.е. определить регистры MCUCR и GIMSK по указанию преподавателя
4. Организовать работу UART по следующей схеме:
    - Инициализировать прерывания по событию завершения приёма;
    - Инициализировать UART для обмена со значениями скорости, тактовой частоты микроконтроллера и форматом данных, заданными преподавателем.
5. Проверить работу программы в симуляторе ASTUDIO.

Контрольные вопросы для самопроверки.

1. Какие регистры используются для работы с параллельными портами?
2. Как запрограммировать линии портов на ввод и вывод?
3. Как программируются входы внешних сигналов прерывания?
4. Форматы посылок UART.
5. Регистр состояния UART.
6. Регистр управления UART.
7. Какие линии порта D используются для работы UART и как они программируются?
8. Как задаётся значение скорости информационного обмена по линиям связи UART.
9. Как инициализируются внешние прерывания в регистрах MCUCR и GIMSK

# Лабораторная работа № 3

## Режимы адресации, работа с регистрами и EEPROM

**Цель работы:** освоить программирование операций пересылки данных с разными режимами адресации и работу с запоминающим устройством EEPROM

### Теоретические сведения

**Файл регистров общего назначения**

На рисунке 2 представлена структура файла регистров общего назначения (РОН).

Каждому регистру соответствует адрес в первых 32 ячейках оперативной памяти данных.

| | 7 … 0 | Адрес | |
|---|---|---|---|
| РЕГИСТРЫ ОБЩЕГО НАЗНАЧЕНИЯ | R0 | $00 | |
| | R1 | $01 | |
| | R2 | $02 | |
| | … | | |
| | R13 | $0D | |
| | R14 | $0D | |
| | R15 | $0F | |
| | R16 | $10 | |
| | R17 | $11 | |
| | … | | |
| | R26 | $1A | Младший байт регистра X (XL) |
| | R27 | $1B | Старший байт регистра X (XH) |
| | R28 | $1C | Младший байт регистра Y (YL) |
| | R29 | $1D | Старший байт регистра Y (YH) |
| | R30 | $1E | Младший байт регистра Z (ZL) |
| | R31 | $1F | Старший байт регистра Z (ZH) |

Рисунок 2 Регистры общего назначения CPU микроконтроллеров AVR.

Последние 6 регистров (R26 и R27, R28 и R29, R30 и R31) могут использоваться как 3 16-ти разрядных регистра – соответственно парам X, Y и Z указанным на рисунке в скобках. Обычно они применяются для хранения адресов памяти.

В командах пересылки данных используются режимы прямой и косвенной адресации. В первом случае в поле операндов указан регистр или адрес ОЗУ, содержащий данные, а во втором регистр, содержащий адрес данных в оперативной памяти.

При работе с регистрами от R0 до R15 используется только косвенная адресация. При работе с регистрами от R16 до R31 используется любая адресация.

**ПРИМЕР 1**

```
ldi    ZH,HIGH(BUFIRS+1)     ;пересылка адреса второй ячейки массива BUFIRS
       ldi    ZL,LOW(BUFIRS+1)      ;в регистр Z в режиме прямой адресации
       ld          R15,Z             ;пересылка байта данных из второй ячейки массива
                                     ;BUFIRS в РОН  R15 в режиме косвенной адресации
       ldi        XH,HIGH(SSS)       ;пересылка адреса ячейки ОЗУ SSS в регистр X
```

```
ldi     XL,LOW(SSS)          ;в режиме прямой адресации
ldi     R10,$0F              ; пересылка байта данных из РОН  R10 в ячейку ОЗУ
st        X,R10              ;SSS в режиме косвенной адресации
lds       R16, BUFIRS+1      ;пересылка байта данных в РОН R16 из ячейки ОЗУ
                             ;с адресом BUFIRS+1
ldi       R16, $0F           ; пересылка байта данных в РОН R16 из ячейки ОЗУ
                             ; с адресом $0F
sts       SSS, R16           ; пересылка байта данных в ячейку ОЗУ с адресом SSS
```

Для обеспечения доступа к записи и чтению EEPROM в контроллере используются три регистра, доступные для чтения и записи:

- 16-ти разрядный регистр адреса EEPROM EEAR (EEPROM ADDRES REGISTER)
- 8-ми разрядный регистр данных EEPROM EEDR (EEPROM DATA REGISTER);
- 8-ми разрядный регистр управления EEPROM EECR (EEPROM CONTROL REGISTER);

EEARH $1F(3F) и EEARL $1E(3E) (соответственно его старший и младший байты).

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Чтение/Запись | R | R | R | R | R | R | R | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Биты | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| $1F ($3F) | - | - | - | - | - | - | - | **EEAR8** |
| $1E($3E) | **EEAR7** | **EEAR6** | **EEAR5** | **EEAR4** | **EEAR3** | **EEAR2** | **EEAR1** | **EEAR0** |
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Этот регистр предназначен для указания адреса от 0 до 511 при чтении данных из EEPROM и их записи в EEPROM.

Регистр данных EEPROM **EEDR** (EEPROM DATA REGISTER). $1D($3D), используемый как буферный при чтении и записи.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| $1D ($3D) | **MSB** | - | - | - | - | - | - | **LSB** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Регистр управления EEPROM **EECR** (EEPROM CONTROL REGISTER).$1C(3C)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| $1D ($3D) | - | - | - | - | - | **EEMWE** | **EEWE** | **EERE** |
| Чтение/Запись | R | R | R | R | R | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТЫ 7-3**: резервные:

**БИТ 2 – EEMWE:** управление записью в EEPROM (EEPROM Master Write Enable):

Если этот бит установлен в состояние лог. 1, и бит EEWE программно установлен в лог. 1 через 4 такта, то данные из регистра данных запишутся по адресу, указанному в регистре адреса.

**БИТ 1 – EEWE:** разрешение записи в EEPROM (EEPROM Write Enable):

Это сигнал записи в EEPROM (строб), во время которого происходит запись байта данных в EEPROM:

Последовательность операций при записи в EEPROM:

1. Ждем, пока бит EEWE очистится аппаратно.

2. Заносим в регистр адреса адрес записываемого байта данных (в младший байт EEARL и в старший байт EEARH).
3. Записываем байт данных в регистр данных EEPROM (EEDR).
4. Устанавливаем бит EEMWE в лог. 1.
5. Через 4 такта временной задержки устанавливаем бит EEWE в лог. 1.

**БИТ 0 – EERE:** разрешение чтения EEPROM (EEPROM Read Enable):
Это строб чтения EEPROM. Сначала в регистр адреса заносится адрес, из которого нужно прочесть байт данных. Прежде чем прочесть данные, необходимо дождаться аппаратной очистки бита EERE, после чего установить его в лог. 1. Когда этот бит очищается, считываемый байт данных из EEPROM переносится в регистр данных EEPROM, после чего его можно переслать в ОЗУ.

**ПРИМЕР 2**

```
;подпрограма записи в EEPROM по адресу в регистре X байта данных из регистра R2
wrd:  sbic   EECR,1          ;ожидание сброса строба записи (готовности к записи)
      rjmp   WRD

      out    EEARL,XL        ;запись адреса в регистр  X
      out    EEARH,XH
      out    EEDR,R2         ;запись байта данных в регистр данных из РОН R2

      sbi    EECR,2          ;установка бита разрешения записи
      nop                    ;задержка на 1 такт (пустая команда – нет операции)
      nop                    ;задержка на 1 такт
      nop                    ;задержка на 1 такт
      nop                    ;задержка на 1 такт
      sbi    EECR,1          ;установка строба записи
      ket                    ;возврат из подпрограммы
```

**ПРИМЕР 3**

```
;подпрограмма чтения из ячейки EEPROM по  адресу указанному в регистре Z в регистр R16
 rrd:  out    EEARL,ZL       ;запись адреса в регистр Z
       out    EEARH,ZH
eer0:  in     R16,EECR       ;пересылка содержимого регистра EECR в РОН R16
       sbrc   R16,0          ;ожидание сброса бита EERE
       rjmp   EER0

       ldi    R16,1          ;установка бита EERE в состояние лог. 1
       out    EECR,R16
eer1:  in     R16,EECR
       sbrc   R16,0                  ;ожидание окончания чтения байта из EEPROM
       rjmp   EER1
       in     R16,EEDR               ;пересылка принятого байта данных из регистра
                                     ;данных EEPROM в РОН R16
       ret                           ;возврат из подпрограммы
```

В основной программе должны быть команды CALL k вызова этих подпрограмм.

## Задание

1.Записать константу, указанную в таблице 3.1 в Н-коде в сегмент EEPROM.

2.Извлечь записанную константу программно и через регистр общего назначения R12 записать в ОЗУ.

3.Разделить константу, помещенную в R12, на число, указанное в таблице 3.1, и полученное значение поместить в новую ячейку EEPROM и другую ячейку ОЗУ, используя регистр R16.

4.Просмотреть работу программы в симуляторе ASTUDIO.

Таблица 3.1. Варианты

| № п/п | Константа в десятичном виде | Делитель |
|---|---|---|
| 1. | 100 | 2 |
| 2. | 160 | 4 |
| 3. | 240 | 8 |
| 4. | 128 | 16 |
| 5. | 64 | 32 |
| 6. | 40 | 2 |
| 7. | 240 | 4 |
| 8. | 160 | 8 |
| 9. | 64 | 16 |
| 10. | 128 | 32 |

Контрольные вопросы для самопроверки

1. Как определяется адрес данных при прямой адресации?
2. Как определяется адрес данных при косвенной адресации?
3. Как осуществляется адресация через регистры X, Y и Z?
4. Для чего нужен регистр данных EEPROM?
5. Как заносится адрес данных в регистр EEPROM?
6. Каково назначение битов регистра управления EEPROM?
7. Как программируется запись данных EEPROM?
8. Как считываются данные из EEPROM?
9. Почему при работе с EEPROM не используются режимы адресации, применяемые при работе с ОЗУ?

# Лабораторная работа № 4
# Таймеры-счетчики и их программирование

**Цель работы:** приобретение практических навыков программирования таймеров/счётчиков

### Теоретические сведения

В контроллере AT90S8515 есть два таймера/счетчика TC0 (Timer/Counter0) и TC1(Timer/Counter1).

**TC0 – 8-разрядный таймер/счетчик**

TC0 имеет два входа: вход импульсов тактовой частоты поступающих с предварительного делителя (режим таймера) и вход, на который подаются импульсы с вывода микроконтроллера – линия 0 порта B (режим счётчика). При переполнении таймера/счётчика TC0 возникает прерывание, если в регистре TIMSK масок прерывания таймеров/счётчиков установлен в состояние лог.

1 бит TOIE0 разрешения прерываний, и устанавливается в состояние лог. 1 флаг INTF0 прерывания в регистре TIFR флагов прерывания таймеров/счётчиков.

Коэффициент деления тактовой частоты CK определяется состояниями бит CS00, CS01 и CS02 регистра управления таймером/счётчиком TC0 **TCCR0** $33 ($53).

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $33 ($53) | - | - | - | - | - | **CS02** | **CS01** | **CS00** |
| Чтение/Запись | R | R | R | R | R | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Состояния бит CS00, CS01, CS02 и соответствующие им режимы приведены в табл. 4.1, где x=0.

Таблица 4.1. Режимы работы таймера/счётчика TC0

| CSx2 | CSx1 | CSx0 | Описание |
|---|---|---|---|
| 0 | 0 | 0 | Стоп таймера1. |
| 0 | 0 | 1 | CK |
| 0 | 1 | 0 | CK/8 |
| 0 | 1 | 1 | CK/64 |
| 1 | 0 | 0 | CK/256 |
| 1 | 0 | 1 | CK/1024 |
| 1 | 1 | 0 | Внешний вход отрицательный фронт |
| 1 | 1 | 1 | Внешний вход положительный фронт |

Результаты счёта находятся в регистре данных (счётном регистре) **TCNT0** таймера/счётчика, доступном для чтения и записи.

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $32 ($52) | **MSB** | - | - | - | - | - | - | **LSB** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**TC1 – 16-ти разрядный таймер/счётчик**

Таймер/счётчик TC1 имеет три входа: вход импульсов тактовой частоты, поступающих с предварительного делителя (режим таймера), вход, на который подаются импульсы с вывода микроконтроллера – линия 1 порта B (режим счётчика) и вход захвата – вывод микроконтроллера ICP.

Кроме режимов таймера и счётчика TC1 поддерживает функции захвата и сравнения и режим широтно-импульсной модуляции (ШИМ).

Режимы работы и функции таймера/счётчика TC1 определяются состояниями бит регистров управления **TCCR1A** и **TCCR1B**.

Регистр **TCCR1A** – адрес $2F($4F)

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $2F ($4F) | **COM1A1** | **COM1A0** | **COM1B1** | **COM1B0** | - | - | **PWM11** | **PWM10** |
| Чтение/Запись | R/W | R/W | R/W | R/W | R | R | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТЫ 7-4 – COM1A1, COM1A0, COM1B1, COM1B0:** управляют режимами работы выходов сравнения **OC1A (линия 5 порта D**) и **OC1B (отдельный вывод)** при совпадении значений чисел в счётном регистре и в соответствующем регистре сравнения **OCR1**A или **OCR1B**.

Описание результатов управления приведено в таблице 4.2, где **X** это **A** (при сравнении с **OCR1A**) или **B (**при сравнении с **OCR1B).**

Таблица 4.2 Управление выходами сравнения

| **COM1X1** | **COM1X0** | **Состояния вывода OC1X** |
|---|---|---|
| 0 | 0 | TC1 отсоединен от вывода **OC1X** |
| 0 | 1 | Переключение (инверсия) выхода **OC1X** |
| 1 | 0 | Сброс **OC1X** в состояние лог. 0 |
| 1 | 1 | Установка **OC1X** в состоянии лог. 1 |

**БИТЫ 1,0 – PWM11, PWM10:** установка режима ШИМ (табл. 4.3).

Таблица 4.3 Режимы ШИМ

| **PWM11** | **PWM10** | **Описание режимов ШИМ** |
|---|---|---|
| 0 | 0 | ШИМ отключена |
| 0 | 1 | Режим ШИМ для 8 разрядов |
| 1 | 0 | Режим ШИМ для 9 разрядов |
| 1 | 1 | Режим ШИМ для 10 разрядов |

В режиме ШИМ TC1 работает как реверсивный счётчик, от нуля до конечного значения в соответствии с выбранной разрядностью ШИМ, после чего счётчик начинает считать в обратную сторону до нуля. Когда значение числа в счётном регистре совпадает с числом в регистре сравнения **OCR1**A или **OCR1B** изменяется состояние соответствующего выхода **OC1A** или **OC1B** согласно установкам битов управления **COM1A1, COM1A0, COM1B1, COM1B0** (табл**.** 4.4).

Таблица 4.4. Реакция на выходе OC1X при совпадении в режиме ШИМ (где X это A или B)

| COM1X1 | COM1X0 | Описание состояния выхода OC1X |
|---|---|---|
| 0 | 0 | Отсоединён |
| 0 | 1 | Отсоединён |
| 1 | 0 | Прямой счёт – лог. 0, обратный – лог. 1. Неинвертирующий ШИМ |
| 1 | 1 | Прямой счёт – лог. 1, обратный – лог. 0. Инвертирующий ШИМ |

Регистр **TCCR1B** – адрес $2E($4E)

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $2E ($4E) | **ICNC1** | **ICES1** | | | **CTC1** | **CS12** | **CS11** | **CS10** |
| Чтение/Запись | R/W | R/W | R | R | R | R | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТ 7 – ICNC1:** подавление шума на входе захвата. Когда этот бит в состоянии лог. 1 функция захвата выполняется, если сигнал захвата повторяется подряд четыре раза, а когда лог. 0 по первому сигналы захвата.

**БИТ 6 – ICES1**: бит выбора фронта сигнала захвата. Когда этот бит в состоянии лог. 0, содержимое счётного регистра ТС1 запоминается в регистре захвата по спадающему фронту сигнала на входе захвата **ICP**, а когда в состоянии лог. 1, по нарастающему фронту.
**БИТ 3 – CTC1**: очистка (сброс) содержимого счётного регистра таймера/счётчика 1 при совпадении значения числа в счётном регистре с числом в регистре сравнения. Если бит CTC1 в состоянии лог. 0, то таймер счетчик продолжает отсчет. В режиме ШИМ этот бит не влияет на работу ТС1.
**БИТЫ 2, 1, 0 – CS12, CS11, CS10**: биты выбора частоты, формируемой таймером/счетчиком 1: Состояния этих битов и соответствующие им режимы работы приведены в табл. 4.1.

Переполнение ТС1, захват и совпадение при сравнении сопровождаются прерываниями, если они разрешены. Маски прерываний таймеров счётчиков ТС0 и ТС1 находятся регистре **TIMSK** по адресу $39($59).

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $39 ($59) | **TOIE1** | **OCIE1A** | **OCIE1B** | **-** | **TICIE1** | **-** | **TOIE0** | **-** |
| Чтение/Запись | R/W | R/W | R/W | R | R/W | R | R/W | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**БИТ 7 – TOIE1**: разрешение прерывания по переполнению таймера/счетчика 1:
Когда этот бит и бит глобального разрешения прерываний в регистре **SREG** установлены в состояние логической единицы, при переполнении ТС1 произойдет прерывание.
**БИТЫ 6,5 – OCIE1A, OCIE1B**: разрешение прерывания по совпадению при выполнении функции сравнения.

Когда этот бит и бит глобального разрешения прерываний в регистре **SREG** установлены в состояние логической единицы, при совпадении значения числа в счётном регистре с содержимым регистра сравнения **OCR1A**, **OCIE1B** генерируется запрос на прерывание.
**БИТ 4** – резервный.
**БИТ 3 – TICIE1**: разрешения прерывания по условию захвата.
Когда этот бит и бит глобального разрешения прерываний в регистре **SREG** установлены в состояние логической единицы при захвате генерируется запрос на прерывание.
**БИТ 2** – резервный.
**БИТ 1 – TOIE0**: разрешение прерывания по переполнению таймера/счетчика 0:
Когда этот бит и бит глобального разрешения прерываний в регистре **SREG** установлены в состояние логической единицы при переполнении ТС0 генерируется запрос на прерывание.
**БИТ 0** – резервный бит:

О том, что условия для прерываний сформированы, сигнализируют логические единицы в регистре флагов этих событий **TIFR** $38($58), биты которого совпадают по номеру с соответствующими им битами в регистре масок прерывания.

| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| $38 ($58) | **TOV1** | **OCF1A** | **OCF1B** | **-** | **ICF1** | **-** | **TOV0** | **-** |
| Чтение/Запись | R/W | R/W | R/W | R | R/W | R | R/W | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Счётный регистр таймера/счётчика 1, регистры сравнения и захвата имеют 16 разрядов.
Счётный регистр состоит из двух восьмиразрядных регистров **TCNT1H** и **TCNT1L**.

| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|---|
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Биты | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $2D ($4D) | **MSB** | - | - | - | - | - | - | - |
| $2C($4C) | - | - | - | - | - | - | - | **LSB** |
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Каждый из регистров сравнения состоит из двух восьмиразрядных регистров **OCR1XH** и **OCR1XL**, где **X –** это **A** или **B**.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Биты | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| $2B ($4B) | **MSB** | - | - | - | - | - | - | - |
| $2A($4A) | - | - | - | - | - | - | - | **LSB** |
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Чтение/Запись | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Так как счётный регистр и регистры сравнения шестнадцатиразрядные, то при записи данных в эти регистры и чтении из них старший и младший байты чисел передаются по очереди. При записи первым передаётся старший байт, а при чтении младший.

Регистр захвата состоит из двух восьмиразрядных регистров **ICR1H** и **ICR1L**,

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Чтение/Запись | R | R | R | R | R | R | R | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Биты | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| $25 ($45) | **MSB** | - | - | - | - | - | - | - |
| $24($44) | - | - | - | - | - | - | - | **LSB** |
| Биты | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Чтение/Запись | R | R | R | R | R | R | R | R |
| Начальное состояние | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**ПРИМЕР**

```
;*****************************
;подпрограмма пуска TC1
;*****************************
pt1:    ldi     R16,$00             ;прерывания не задействованы
        out     TIMSK,R16
        ldi     R16,$00             ;выводы OC1X отключены, ШИМ отключена
        out     TCCR1A,R16
        ldi     R16,$00
        out     TCNT1H,R16          ;очистка счётного регистра
        out     TCNT1L,R16
        ldi     R16,$05             ;захват работает по заднему фронту сигнала,
        out     TCCR1B,R16          ;запуск CT1,  частота CK/1024

        ret                         ;выход из подпрограммы
```

## Задание

1. Задействовать прерывание CT1 по варианту, указанному в табл. 4.5. В подпрограмме обслуживания прерывания остановить CT1.
2. Запретить прерывания, остановить CT1, очистить счётный регистр, записать в регистр сравнения произвольное число, активизировать указанный выход, разрешить прерывания и запустить CT1 с заданной тактовой частой без сброса при совпадении.
3. Проверить работу программы в симуляторе ASTUDIO и сравнить содержимое счётного регистра и регистра сравнения.

Таблица 4.5. Варианты заданий

| Вариант | Прерывание таймера | | | Активизируемый выходной пин | | Пуск таймера/счётчика 1 с тактовой частотой: |
|---|---|---|---|---|---|---|
| | По переполнению | По сравнению с A | По сравнению B | OC1A | OC1B | |
| 1 | | + | | + | | СК/8 |
| 2 | | | + | | + | СК/1024 |
| 3 | + | | | | | СК/64 |
| 4 | | + | | + | | СК/256 |
| 5 | | | + | | + | СК |
| 6 | + | | | | | СК/8 |
| 7 | | + | | + | | СК/256 |
| 8 | | | + | | + | СК/64 |
| 9 | + | | | | | СК |
| 10 | | + | | + | | СК/1024 |

Контрольные вопросы для самопроверки

1. В каких режимах может работать таймер/счётчик 0?
2. Какие входы и выходы есть у таймера/счётчика 0?
3. Как инициализировать прерывания по переполнению таймера/счётчика 0?
4. Как реализуется функция захвата таймера/счётчика 1?
5. Для каких целей применяются функции захвата и сравнения?
6. Как реализовать работу с прерываниями по совпадению при сравнении?
7. Как осуществляются чтение и запись данных для шестнадцатиразрядных регистров?
8. Как работает таймер/счётчик 1 в режиме ШИМ?
9. Как программируются функции захвата и сравнения?
10. Чем отличаются режим таймера и режим счётчика и для чего они применяются?

# Лабораторная работа № 5
# Отладка программ с использованием резидентных средств микроконтроллера 87C51FA

**Цель работы:** ознакомление с архитектурой микроконтроллера 87C51FA и принципами его программирования

## Теоретические сведения

### Описание макета

Лабораторный макет выполнен на базе однокристальной микроЭВМ семейства MCS51. Память программ содержит ПЗУ объемом 16К (адреса 0H … 3FFFH) с резидентными программами «монитор», «редактор», «ассемблер» и сервисными подпрограммами.

Память данных объемом 32К (0H ... 7FFFH) разделена на две части: первая (0H ... 3FFFH) предназначена для редактора и ассемблера, вторая совмещена с памятью программ (4000H ... 7FFFH) и в нее можно помещать как данные, так и исполняемый программный код.

В состав макета входят два цифроаналоговых преобразователя (ЦАП). Первый подключен к порту P1 микроконтроллера, второй – к шине данных, запись в него осуществляется аналогично обращению к ячейке ОЗУ с адресом 8000H.

Макет также содержит перестраиваемый генератор частоты, управляемый формирователь постоянного напряжения $E_{ВХ}$, аналоговый компаратор, выход которого подключен к входу порта P3.3 (INT1), и фильтр нижних частот (ФНЧ), подключенный к выводу P1.5.

**Функции и директивы программы «монитор» (запускается программой ESC1 или ESC2)**

Программа «монитор» предназначена для организации интерфейса пользователь – микроконтроллер. При включении питания или нажатии кнопки «сброс» программируется последовательный порт микроконтроллера и выводится сообщение 'MCS-87C51FA SCS-MONITOR'.

Монитор выполняет одну из директив: D, M, T, C, J, E, F, S в следующих областях памяти: внутреннем ОЗУ (M), внешнем ОЗУ (R) и ПЗУ (P). Символы-указатели области памяти (M, R, P, MM, RR, MR, RM) вводятся вслед за символом директивы без пробела. Директивы могут содержать до трех параметров, представляющих собой числа в шестнадцатеричной системе счисления, разделенные между собой символами <,> или <пробел>. Незначащие нули и символ «H» вводить не нужно. Ввод директивы заканчивается нажатием клавиши <CR> или <ВК>. В случае ошибки при вводе директива не выполняется, выводится символ <#> и директиву необходимо ввести заново.

Примеры директив:

J4000 – запуск программы с адреса 4000H.

DM – просмотр внутреннего ОЗУ (0 … 0DFH).

DR0 3FF – просмотр внешнего ОЗУ с адреса 0 по 3FFH.

DP2000 20FF – просмотр ПЗУ.

TRM7000 7007 3F – пересылка из внешнего ОЗУ блока с адреса 7000H по адрес 7007H во внутреннее ОЗУ, начиная с адреса 3FH.

TMM20 33 80 – то же во внутреннем ОЗУ.

TRR100 200 6000 – то же во внешнем ОЗУ.

CMR30 3F 7000 – сравнение блоков внешнего и внутреннего ОЗУ.

CMM0 F 20 – сравнение областей внутреннего ОЗУ.

FM0 3F 55 – заполнение внутренней памяти кодом 55H.

FR4000 7FFF 55 – то же с внешней памятью.

MM5 – редактирование ячейки 5 внутреннего ОЗУ. При нажатии <пробел> – редактирование следующей ячейки, <CR> – выход из режима.

MR7F00 – то же во внешнем ОЗУ.

S5 – вывод кода символа <5> (справочная директива).

E – вход в программу редактора и транслятора ассемблера MCS51.

При выводе информации, не помещающейся на один экран, остановка/возобновление вывода осуществляется нажатием клавиш <Ctrl>+<S>. Прекращение вывода с выходом в монитор – <Ctrl>+<A>.

**Полноэкранный редактор**

Переход из монитора в режим ввода текста программы осуществляется по директиве <E>. Функции клавиш аналогичны редактору Volkov Commander.

Клавиши <Home> и <End> объединены в одну <Home>.

<BS> или <Ctrl>+<H> – удаление символа слева от курсора.

<Ctrl>+<G> – удаление символа над курсором.

<Ctrl>+<Y> – удаление строки, отмеченной курсором.

<Ctrl>+<K> – очистка с позиции курсора до конца строки.

<ESC> – выход из редактора.

Введенный с клавиатуры текст располагается во внешнем ОЗУ с адреса 0B01H по 3FFFH. Максимальное число строк – 1024. Во избежание потери текста отлаживаемой программы не рекомендуется использовать данный диапазон адресов при трансляции.

Текст из файла загружается программой ESC1 или ESC2.

**Резидентный ассемблер**

Ассемблер преобразует текстовый файл, созданный редактором, в исполняемый машинный код и размещает его по адресу, указанному директивой ORG (не менее 4000H). Трансляция производится до обнаружения в тексте программы директивы END и завершается выводом на экран листинга программы. При обнаружении транслятором ошибки вывод листинга прекращается, выводится сообщение о типе ошибки и при нажатии на любую клавишу вывод листинга продолжается. После завершения трансляции на экран выводится количество обнаруженных ошибок и происходит выход в редактор с установкой курсора в строку, содержащую первую ошибку. Выход из редактора в монитор – клавишей <ESC>.

Предусмотрено использование стандартных директив резервирования памяти DS, DB и присвоения значений константам EQU. Допустимая длина меток и имен – до 6 символов.

Для того чтобы обеспечить выход в монитор по окончании выполнения программы, ее следует завершить командой JMP 0 или RET.

Запуск транслированной программы осуществляется директивой монитора J<adress>.

**Сервисные подпрограммы**

Сервисные подпрограммы в основном обеспечивают реализацию типовых процедур вывода информации на экран дисплея. Резидентный транслятор воспринимает следующие символические имена сервисных подпрограмм:

CI – ввод символа в аккумулятор (ACC) с клавиатуры;

CO – вывод символа из ACC на дисплей;

ACCBIN – вывод (ACC) на дисплей в двоичном виде;

ACCDEC – вывод (ACC) на дисплей в десятичном виде;

ACCHEX – вывод (ACC) на дисплей в HEX-виде (ACC не сохраняется);

DPTRH – вывод (DPTR) на дисплей в HEX-виде (ACC не сохраняется);

DPTRD – вывод (DPTR) на дисплей в десятичном виде;

CRLF – перевод строки на экране;

TXTOUT – вывод на дисплей текста. Адрес первого символа должен быть помещен в DPTR (код 0 – признак конца текста, ACC и DPTR не сохраняются);

OUTMEM – вывод на дисплей в десятичном виде двухбайтного числа, расположенного в ячейках 31H, 30H, и перевод строки;

DAC2 – вывод (ACC) по адресу ЦАП2 (8000H);

DSEC – программная задержка на 0.1 с.

В ПЗУ расположены массивы данных. Адреса первых элементов массивов обозначены символическими именами (метками):

TABSIN – 256 точек одного периода синусоиды (начало массива – 2000H);

TABMOD – 256 точек четырех периодов модулированной синусоиды (начало массива – 2300H).

## Задание

1) Изучить описание лабораторной установки.

2) Освоить практически выполнение директив монитора при операциях с различными типами памяти. Просмотреть области внутреннего и внешнего ОЗУ, ПЗУ (2000H – TABSIN), очистить области внешнего и внутреннего ОЗУ, модифицировать содержимое ОЗУ, ввести текущую дату и номер группы. Размножить введенную информацию (директивой T).

3) Ввести текст и запустить программу:

```
      ORG       4000H
M:    SETB      P1.0
      CLR       P1.0
      ***               ; место анализируемой команды
      JMP       M
      END
```

Проанализировать с помощью осциллографа время выполнения различных групп команд. Перечень команд приведен в Приложении.

4) Написать программу для вывода в центре экрана сообщения:

```
==================
= MICROPROCESSOR =
==================
```

*Указание: использовать подпрограмму TXTOUT и директиву DB.*

**ПРИМЕР**

```
      MOV       DPTR,#TXT
      CALL      TXTOUT
        ***
TXT:  DB        0DH,0AH,'ASCII  SYMBOLS',0
```

5) Написать программу циклического вывода на экран даты и номера группы с помощью подпрограммы CO.

*Указание: использовать подпрограммы CRLF, DSEC. Коды символов определить с помощью директивы S.*

6) Написать программу вывода на экран содержимого аккумулятора в десятичном, двоичном и шестнадцатеричном виде. Содержимое аккумулятора изменять в цикле от нуля до 255. Перевод строки (CRLF) при выводе осуществлять через 1 с. Разделить данные в строке путем вывода символа горизонтальной табуляции.

Отчет по лабораторной работе должен содержать:

- Тексты программ с комментариями.
- Времена выполнения различных групп команд.
- Таблица чисел в DEC-, HEX- и BIN-системах счисления (16 строк).

Контрольные вопросы для самопроверки

1. Какие технические средства содержит макет лабораторной установки?
2. Какие директивы и функции реализуются в программе «монитор»?
3. Как осуществляются редактирование, трансляция и запуск программы?
4. Какие сервисные подпрограммы есть в лабораторной установке?
5. Как просмотреть содержимое ОЗУ и ПЗУ?
6. Как определить время выполнения команды?
7. Как выводятся на экран текстовые сообщения?
8. Как вывести данные в разных системах счисления?

## Лабораторная работа 6
## Работа с данными запоминающих устройств микроконтроллера 87C51FA

**Цель работы:** исследование программной организации памяти микроконтроллера.

## Теоретические сведения

Порядок вычисления исполнительного адреса для команды определяет способ адресации. В командах обработки данных исполнительный адрес является адресом данных, а в командах перехода – адресом, по которому команда передает управление. Способы адресации делятся на прямые и косвенные. В первом случае исполнительный адрес является частью команды или вычисляется с использованием содержимого одного из регистров и указанного в команде смещения. При косвенной адресации в команде содержится адрес ячейки памяти, в которой записан исполнительный адрес.

В однокристальной микроЭВМ 87С51 используется 5 способов адресации: регистровая, прямая, косвенно-регистровая, непосредственная и косвенно-регистровая по сумме базового и индексного регистров. Эти способы используются в различных сочетаниях и обеспечивают 21 режим адресации.

**Регистровая адресация** используется для обращения к восьми регистрам выбранного банка, сдвоенному регистру DPTR, регистрам A и B (MOV A,R*n*; XCH A,R*n*; DIV A,B и т. п.). Регистровая адресация позволяет получать однобайтовый эквивалент двухбайтовых команд прямой адресации.

**Прямая адресация** используется для обращения к нижней половине внутреннего ОЗУ (адреса 0 ... 7FH) и к регистрам специального назначения, расположенным по адресам 80H ... FFH. Регистры выбранного банка могут быть адресованы обоими способами, например, при выборе нулевого банка команды MOV R0,A и MOV 0,A выполняют одно и то же действие – пересылку содержимого аккумулятора в регистр R0, который располагается в памяти по адресу 0H. Команда MOV R0,30H использует оба рассмотренных способа адресации, пересылая в регистр R0 содержимое ячейки памяти, расположенной по адресу 30H. Символические имена регистров специального назначения, используемые в командах (ACC, PSW, SP, TCON, PCON и др.), при ассемблировании программы автоматически заменяются прямыми адресами.

Прямая адресация используется также для обращения к 256 отдельно адресуемым битам, 128 из которых расположены в ячейках внутреннего ОЗУ с адресами 20H ... 2FH, а остальные – в регистрах специального назначения (регистры-защелки портов, ACC, B, PSW, регистры управления системой прерывания и таймерами микроЭВМ). При обращении к биту используется символическое имя бита в регистре (SETB ACC.2, CLR PSW.3, CPL P1.3, CLR C) или его прямой адрес (SETB 00H, ANL C,0FFH, CPL 80H и т. п.). Бит с адресом 00H является младшим битом ячейки памяти с адресом 20H, бит FFH – старшим битом ячейки с адресом 2FH.

**Косвенно-регистровая адресация** позволяет обращаться к внутренней и внешней памяти данных (ОЗУ). Символ косвенной адресации – @. В качестве регистров указателей, содержащих адрес, по которому производится обращение, используются регистры R0 или R1 выбранного банка регистров. MOV A,@R1 – команда пересылки в аккумулятор содержимого ячейки внутреннего ОЗУ, адрес которой хранится в регистре R1.

Обращение к внутреннему ОЗУ с адресами 80H ... FFH возможно только с помощью косвенной адресации, поскольку такие же адреса имеют регистры специального назначения, доступные только с помощью прямой адресации.

Обмен между микроЭВМ и внешним ОЗУ возможен только через аккумулятор с помощью косвенной адресации. Для обращения к любой из 64К ячеек в качестве указателя адреса используется шестнадцатиразрядный регистр DPTR (MOVX A,@DPTR; MOVX @DPTR,A). Возможен обмен с внешним ОЗУ в пределах блока из 256 ячеек (страницы) с использованием R0, R1 (MOVX A,@R*i*; MOVX @R*i*,A). В этом случае номер страницы (старший байт адреса) задается содержимым порта P2. Этот вариант может применяться в конфигурациях, использующих только внутреннее ПЗУ, так как при считывании очередной команды из внешнего ПЗУ в порт P2 выводится старший байт адреса команды.

Косвенно-регистровую адресацию используют команды PUSH и POP. В качестве указателя адреса выступает восьмиразрядный регистр – указатель стека (SP). Стек в микроЭВМ семейства MCS51 расположен во внутреннем ОЗУ. По сигналу RESET в SP записывается адрес 07H, т. е. под стек отведено все внутреннее ОЗУ, кроме нулевого банка регистров. Программа «монитор» переназначает стек с адреса E0H.

**Непосредственная адресация** позволяет использовать константы, явно указанные в команде. Символ непосредственной адресации – #. Например, MOV DPTR,#5000H – команда загрузки указателя адреса, SUBB A,#230 – вычитание из аккумулятора числа 230 и т. п.

**Косвенно-регистровая адресация по сумме:** базовый регистр плюс индексный регистр (содержимое аккумулятора A) упрощает просмотр таблиц, зашитых в памяти программ. Любой байт из таблицы может быть выбран по адресу, определяемому содержимым DPTR или PC и содержимым аккумулятора, затем помещен в аккумулятор (команды MOVC A,@A+DPTR и MOVC A,@A+PC). Ниже приведен фрагмент программы, использующей косвенно-регистровую адресацию по сумме для вывода в порт P1 одного периода синусоидального сигнала, 256 отсчетов которого записаны в таблице с адреса TABSIN.

**ПРИМЕР**

```
      MOV     DPTR,#TABSIN     ;загрузка базового регистра
      MOV     20H,#0           ;загрузка счетчика отсчетов
M:    CLR     A                ;очистка индексного регистра
      MOVC    A,@A+DPTR        ;выборка очередного отсчета
      MOV     P1,A
      INC     DPTR             ;увеличение базового адреса на единицу
      DJNZ    20H,M            ;условный переход по содержимому 20H
```

## Задание

1) Ознакомиться с описанием.

2) Получить вариант индивидуального задания по разделам заданий **A** и **B,** приведенным ниже.

3) Составить и отладить программу, проверить результаты ее работы на лабораторном макете, используя директивы монитора (DM, DR).

**Варианты индивидуальных заданий:**

**A.** Создать с помощью директивы MM монитора во внутреннем ОЗУ массив из натурального ряда чисел от 0 до 21, начиная с адреса 30H. Отладить и запустить программу, с помощью которой:

1. Обнулить все ячейки памяти, содержащие нечетные элементы массива. Подсчитать количество оставшихся ненулевых элементов и вывести это число на экран.

2. Обработать «разреженный» массив, содержащий нулевые элементы, и переслать обнаруженные ненулевые элементы во внутреннюю и во внешнюю память, начиная с адресов 60H и 5000H соответственно.

*Указание: перед запуском программы с помощью директивы MM обнулить некоторые элементы массива.*

3. Переписать элементы массива, превысившие порог, в память с адреса 50H. Подсчитать количество превышений порога и вывести результат на дисплей.

*Указание: перед запуском программы с помощью директивы MM занести в ячейку 60H код порога (1 ... 20).*

4. Вывести на дисплей адреса ячеек памяти, содержащих элементы массива, совпадающие с эталоном, хранящимся в ячейке с адресом 50H. *Указание: перед запуском программы с помощью директивы MM ввести код эталона в пределах 1 ... 20.*

5. Вывести на дисплей значения и адреса наибольшего и наименьшего элементов в массиве. Сохранить значения в ячейках 10H, 11H.

6. Вывести на дисплей элементы массива, которые укладываются в границы допуска. Отдельной строкой вывести количество выведенных элементов.

*Указание: перед запуском программы в ячейках 10H и 11H расположить коды нижней и верхней допусковых границ.*

7. Переслать во внешнюю память, начиная с адреса 5000H, все элементы массива, кратные 3, с адреса 5030H – кратные 4.

8. Вычислить и вывести на дисплей среднее арифметическое всех элементов массива. Записать его в ячейку 5000H.

**Б.** В памяти программ макета с адреса TABSIN расположен массив из 256 чисел, представляющих один период синусоидального колебания. Нулевому уровню соответствует код 128 (80H). Отладить и запустить программу, с помощью которой:

1. Подсчитать количество четных и нечетных элементов массива. Вывести результаты на дисплей.

2. Переписать во внутреннее ОЗУ с адреса 20H только четные элементы массива.

3. Переписать в ОЗУ с адреса 20H элементы массива, лежащие ниже нулевого уровня. Подсчитать их количество и вывести результат на экран.

4. Переписать элементы массива, оказавшиеся ниже порога, в память с адреса 20H и вывести на дисплей их количество.

*Указание: перед запуском программы с помощью директивы MM ввести в ячейку 10H код порога (10 … 120).*

5. Вывести на дисплей адреса совпавших с эталоном элементов массива. Сохранить адреса во внутреннем ОЗУ, начиная с ячейки 20H.

*Указание: перед запуском программы с помощью директивы MM ввести в ячейку 10H код эталона (10 … 250).*

6. Вывести на дисплей значения и адреса минимального и максимального элементов в массиве. Сохранить значения элементов во внутреннем ОЗУ в ячейках 60H и 61H.

7. Записать в ОЗУ с адреса 20H все элементы массива, кратные 5. Подсчитать их количество и вывести результат на экран.

8. Вывести на дисплей количество элементов массива, которые укладываются в границы допуска.

*Указание: в ячейках 60H и 61H с помощью директивы MM расположить числа, имитирующие нижнюю и верхнюю допусковые границы.*

Отчет по лабораторной работе должен содержать:

- Вариант задания.
- Тексты программ с комментариями и указанием используемых видов адресации.

Контрольные вопросы для самопроверки

1. Чем отличаются режимы регистровой и косвенно-регистровой адресации?
2. Как реализуется прямая адресация?
3. Что представляет собой режим непосредственной адресации?
4. Какую роль при адресации памяти исполняет регистр DPTR?
5. Какие имена регистров специальных функций, используются в командах?
6. Как создать в ОЗУ с помощью директивы монитора MM массив чисел?
7. Как переслать данные из внутренней памяти во внешнюю?
8. Как вычислить среднее значение элементов массива чисел?
9. Как определить количество элементов массива, числовые значения которых находятся в пределах допуска?

# Лабораторная работа 7

# Программирование портов микроконтроллера 87C51FA и формирование аналоговых сигналов

**Цель работы:** исследование методов программирования параллельных портов и формирования аналоговых сигналов с помощью микроконтроллера.

## Теоретические сведения

### Порты ввода/вывода

МикроЭВМ семейства MCS51 обладают разнообразными возможностями для обмена информацией с внешней средой. Ввод/вывод цифровых данных может осуществляться последовательно и параллельно. При первом способе используется последовательный порт (последовательный интерфейс), который программируется в один из четырех режимов, поддерживающих стандартные протоколы обмена (например, RS232C). Скорость обмена задается программой.

Параллельный ввод/вывод производится через двунаправленные порты P0...P3. Каждый порт содержит восьмиразрядный регистр, имеющий байтовую и битовую адресации.

Наличие битовой адресации позволяет программировать любую из линий, независимо от других линий порта (SETB P1.2; CLR P3.5; CPL P2.7; MOV P3.5,C и т. п.). Состояние на соответствующем входе порта (ввод бита) может быть скопировано во флаг переноса или проверено командами условного перехода (MOV C,P3.4; JB P0.6,LABEL; JNB P0.4,LABEL).

Регистры параллельных портов могут быть регистрами назначения или операндами в командах. В первом случае данные записываются в регистр порта (MOV P1,A; INC P1; и т. п.), во втором – в качестве операнда выступает состояние входов микроЭВМ (ORL A,P2; MOV R3,P0). Для правильного определения логического уровня на входной линии порта в соответствующем разряде регистра порта должна быть записана логическая «1». При этом закрыт транзистор, формирующий уровень «0», и потенциал входа подтянут к «1» через внутренний (порты P1 – P3) или внешний (порт P0) резистор с номиналом порядка 10 кОм. Если выходной транзистор линии порта открыт, то источник цифровой информации не в состоянии сформировать на входе уровень «1» и состояние линии ввода воспринимается как «0».

По сигналу RESET в регистры P0 ... P3 записывается код FFH, т. е. все порты настраиваются на ввод информации.

Каждый из портов микроЭВМ кроме функций ввода/вывода выполняет альтернативные функции. При работе с внешней памятью (команды MOVX и MOVC) порт P0 выводит младший байт адреса (A0 ... A7), производит ввод/вывод данных (D0 ... D7). Адрес/данные мультиплексированы во времени. Порт P2 выводит старший байт адреса (A8 ... A15).

Линии порта P1 отведены под управление таймером T2 и массивом программируемых счетчиков (PCA).

Порт P3 используется различными системами микроЭВМ:

| | | | |
|---|---|---|---|
| P3.0 | Вход | Приемник последовательного порта | RxD |
| P3.1 | Выход | Передатчик последовательного порта | TxD |
| P3.2 | Вход | Внешнее прерывание | INT0 |
| P3.3 | Вход | Внешнее прерывание | INT1 |
| P3.4 | Вход | Таймер / счетчик | T0 |
| P3.5 | Вход | Таймер / счетчик | T1 |
| P3.6 | Выход | Строб сигнала записи во внешнее ОЗУ | WR |
| P3.7 | Выход | Строб сигнала чтения из внешнего ОЗУ | RD |

Альтернативные функции любой из линий порта P1 и P3 реализуются в том случае, если в соответствующем разряде регистра порта содержится «1». В противном случае на выходе линии будет присутствовать «0».

При выполнении команд MOVX и MOVC содержимое регистра P2 не модифицируется, а в регистр P0 код FFH записывается аппаратно.

Описанные аппаратные средства микроЭВМ позволяют поддерживать практически любой из стандартных протоколов обмена цифровыми данными, а также создавать специализированные интерфейсы.

**Формирование аналоговых сигналов**

Цифровая информация передается прямоугольными видеоимпульсами. Спектр такого сигнала содержит постоянную составляющую и медленно убывает с ростом частоты по закону $\sin(x)/x$. Передача прямоугольных импульсов по узкополосным каналам связи приводит к искажениям их формы и ошибкам на приемной стороне. Стандартным методом уменьшения занимаемой полосы и, соответственно, искажений при передаче цифровой информации является использование узкополосных сигналов-переносчиков, параметры которых модулируются (манипулируются) цифровыми сигналами. К основным параметрам относятся амплитуда, фаза и частота. При двоичной манипуляции одного параметра сигнал переносит один бит информации, при манипуляции двух параметров (амплитуда и фаза, амплитуда и частота) элементарный сигнал переносит два бита. Для еще большего информационного уплотнения используется многопозиционная модуляция параметра (4 – 8 положений фазы несущей, до 4 уровней амплитуды огибающей, 8 – 16 позиций по частоте). Количество многопозиционных сигналов ограничено различимостью сигналов с соседними позициями на приемной стороне. Сигналы с частотой дискретизации до 20 кГц могут быть сформированы средствами микроЭВМ.

Методы формирования основаны на представлении сигнала совокупностью выборочных значений, хранящихся в ПЗУ, которые последовательно выводятся на ЦАП. Объем выборки определяется частотой дискретизации, достаточной для восстановления непрерывного сигнала интерполирующим фильтром. Разрядность выборочных значений определяет точность формирования амплитуды и ограничена разрядностью ЦАП.

В соответствии со значением текущего модулирующего бита программа формирования выводит записанный в ПЗУ сигнал без изменений или перед выводом производит модификацию очередной выборки.

Амплитудная манипуляция производит пропорциональное уменьшение кода сигнала. Двухпозиционная фазовая манипуляция соответствует изменению знака сигнала, что сводится к инверсии кода. При частотной манипуляции следует запретить вывод в ЦАП некоторого числа выборок, что эквивалентно уменьшению длины массива и дискретному увеличению частоты.

Ниже приведен пример подпрограммы формирования сигнала с линейно-частотной модуляцией. Для наглядности частота каждого следующего периода синусоиды удваивается. Число периодов сигнала задается в ячейке 30H.

```
LFM:    MOV     DPTR,#TABSIN
        MOV     R7,#1
        MOV     R6,#255
K:      CLR     A
        MOVC    A,@A+DPTR
        MOV     P1,A
        MOV     A,DPL
        ADD     A,R7
        MOV     DPL,A
        DJNZ    R6,K
```

```
MOV     A,R7
ADD     A,R7
MOV     R7,A
MOV     A,#255
MOV     B,R7
DIV     AB
MOV     R6,A
DJNZ    30H,K
RET
```

В ПЗУ макета записаны два массива выборок по 256 отсчетов каждая. TABSIN – период синусоиды, TABMOD – 4 периода синусоидальной несущей с огибающей, сглаженной по закону квадрата синуса. Нулевому уровню сигналов соответствует код 128 (80H).

## Задание

1) Ознакомиться с теоретической частью работы.

2) Отладить и запустить циклическую программу анализа состояния входов порта P3. Произвести в программе инкремент ячейки 30H при наличии «0» на входе P3.3 и декремент при наличии «0» на входе P3.2. Вывести содержимое ячейки 30H на экран в десятичном коде с помощью подпрограмм ACCDEC, CRLF и DSEC. Проверить работу программы, кратковременно подключая P3.3 и P3.2 к шине «Общий». Модифицировать программу, добавив вывод в P3.2 инвертированного состояния входа P3.3. Подключать к шине «Общий» только вход P3.3.

3) Отладить и запустить циклическую программу вывода в порт P1 «бегущей единицы». Направление сдвига задавать входом P3.2. Управлять частотой сдвига входом P3.5, обнаруживая перепад 1/0. Подключить к P3.5 сигнал генератора макета. Наблюдать осциллограммы на выводах порта P1 и выходе ЦАП1, меняя частоту генератора и сигнал на входе P3.2.

4) Составить программу формирования на выходе ЦАП1 частотно-манипулированного сигнала с одинаковой длительностью посылок «0» и «1». Использовать массив TABSIN. Логический «0» передавать одним периодом синусоиды, логическую «1» – двумя. Модулирующую информацию вводить, последовательно опрашивая входы P3.2 ... P3.5. Соединить один или два входа с шиной «Общий». Наблюдать сигнал на выходе ЦАП1.

5) Составить программу формирования фазоманипулированного сигнала на выходе ЦАП1. В качестве элементарной посылки сигнала использовать массив TABMOD. Модуляция – через входы P3.2 ... P3.5.

6) Составить программу формирования амплитудно-манипулированного сигнала с соотношением уровней амплитуд огибающей сигнала 1 : 2. В качестве элементарной посылки сигнала использовать массив TABMOD. Управление модуляцией – через входы P3.2 ... P3.5.

7) Вызвать в бесконечном цикле подпрограмму формирования сигнала с линейно-частотной модуляцией (CALL LFM). В ячейку 30H занести коэффициент изменения частоты формируемого сигнала (3 ... 9). Наблюдать осциллограмму на выходе ЦАП1. Написать комментарии к подпрограмме LFM.

Отчет по лабораторной работе должен содержать:

- Тексты программ с комментариями.
- Осциллограммы сигналов.

Контрольные вопросы для самопроверки

1. Какие порты ввода/вывода есть у микроконтроллера?
2. Как программируются линии параллельных портов на ввод и вывод данных?
3. Какие альтернативные функции выполняют линии параллельных портов?
4. Какие виды узкополосных сигналов используются в цифровых каналах связи?
5. Что содержится в массивах TABSIN и TABMOD?

6. Как сформировать на выходе ЦАП частотно-манипулированный сигнал?
7. Как сформировать на выходе ЦАП фазо-манипулированный сигнал?
8. Как сформировать на выходе ЦАП амплитудно-манипулированный сигнал?
9. Как сформировать на выходе ЦАП сигнал с линейно-частотной модуляцией?

# Лабораторная работа 8
# Аналого-цифровой преобразователь сигналов на базе микроконтроллера 87C51FA

**Цель работы:** исследование принципов и характеристик аналого-цифрового преобразования сигналов с помощью микроконтроллера.

### Теоретические сведения

Задачей аналого-цифрового преобразователя (АЦП) является квантование процесса с непрерывными значениями по времени (дискретизация) и по уровню.

Дискретизацию выполняет устройство выборки-хранения (УВХ), которое должно обеспечить на входе АЦП постоянство измеряемого отсчета сигнала во время преобразования с точностью не ниже младшего разряда шкалы АЦП. Динамическую погрешность, возникающую при отсутствии УВХ, можно оценить соотношением $E = v_M T_P$, где $v_M$ – максимальная скорость изменения сигнала, а $T_P$ – время преобразования.

Точность АЦП принято оценивать значением среднеквадратичной погрешности. Максимальная погрешность обычно принимается равной единице младшего разряда шкалы АЦП. Влияние шумов на точность преобразования может быть существенно уменьшено путем статистической обработки результатов. Нелинейность преобразователя характеризуется идентичностью приращений кодовых эквивалентов при фиксированных приращениях сигнала во всем диапазоне преобразования. Дифференциальная нелинейность характеризует локальную погрешность при формировании соседних значений кода.

Многие выпускаемые промышленностью микросхемы АЦП реализуют преобразование напряжения в код методом поразрядного уравновешивания. Метод основан на последовательном сравнении преобразуемого напряжения с набором эталонных уровней, формируемых ЦАП. Сравнивающее устройство (компаратор) вырабатывает цифровой сигнал «1» или «0», отражающий результат сравнения. Набор эталонных уровней представлен рядом: $U_1 = E/2$, $U_2 = C_1E/2 + E/4$, $U_3 = C_1E/2 + C_2E/4 + E/8$, и т.д., где $E$ – опорное (эталонное) напряжение ЦАП, а $C_k$ – выходной сигнал компаратора на $k$-м шаге сравнения. Типичные микросхемы АЦП имеют $T_P = 1 \ldots 10$ мкс при разрядности 8 ... 12 бит, т. е. затраты времени на обслуживание (запуск АЦП и ввод данных в микропроцессор) соизмеримы с $T_P$. При отсутствии жестких требований по частоте дискретизации аналого-цифровое преобразование с приемлемыми характеристиками можно реализовать, как показано ниже, на основе программного управления ЦАП, дополненного компаратором.

Для любого АЦП, содержащего ЦАП, минимальное время, необходимое для реализации одного шага сравнения $T_J = T_D + T_C$, где $T_D$ – время установления сигнала на выходе ЦАП с заданной точностью; $T_C$ – задержка срабатывания компаратора.

Подпрограмма АЦП, составленная для управления ЦАП, с $T_D < 1$ мкс введена в ПЗУ макета и имеет вид:

```
SADC:MOV     P1,#80H
     JNB     INT1,SB6
     CLR     P1.7
SB6: SETB    P1.6
     JNB     INT1,SB5
     CLR     P1.6
```

```
SB5:    SETB    P1.5
        ***                     ;аналогично для остальных разрядов
SB0:    SETB    P1.0
        JNB     INT1,SB_RET
        CLR     P1.0
SB_RET: RET
```

Для надежной работы АЦП следует обеспечить задержку опроса состояния выхода компаратора. Менее эффективная по скорости, но более короткая в написании подпрограмма АЦП приведена ниже:

```
FADC:   MOV     P1,#128
        MOV     A,P1
        CLR     C
FSC:    ORL     P1,A
        NOP
        NOP
        JNB     INT1,FNR
        XRL     P1,A
FNR:    RRC     A
        JNC     FSC
        RET
```

В одноканальных системах часто используют следящий метод аналого-цифрового преобразования. Однократное преобразование производит инкремент или декремент текущего кодового эквивалента до изменения состояния выхода компаратора. Текст подпрограммы следящего АЦП:

```
SLADC:  MOV     C, INT1
        JC      MNS
PLUS:   INC     P1
        NOP
        ORL     C,INT1
        JNC     PLUS
        RET
MNS:    DEC     P1
        NOP
        ANL     C,INT1
        JC      MNS
        RET
```

## Задание

1) Ознакомиться с теоретической частью работы.

2) Изучить динамические свойства ЦАП. Ввести и запустить программу, циклически формирующую в порте P1 главный кодовый переход 10000000 → 01111111. На инвертирующий вход компаратора подать постоянный уровень напряжения с потенциометра. Наблюдать переходный процесс на выходе ЦАП и компаратора при различных значениях $E_{BX}$.

3) Исследовать аналого-цифровое преобразование по методу поразрядного уравновешивания. Ввести и запустить программу:

```
NN:     MOV     B,#0
        MOV     R1,#20H
        MOV     DPH,#80H
        MOV     R7,#32
```

```
SS:     MOV     R6,#8
        MOV     A,B
        MOVX        @DPTR,A
        ADD     A,#16       ; приращение на выходе ЦАП2
        MOV     B,A
VV:     CALL    FADC;
        MOV     A,P1
        MOV     @R1,A
        INC     R1
        DJNZ    R6,VV
        DJNZ    R7,SS
        JMP     NN
```

Устанавливая на входе IN2 уровень преобразуемого напряжения потенциометром $E_{ВХ}$ (минимальный, средний, максимальный), зарисовать осциллограммы на выходе ЦАП1. Измерить время преобразования.

Подключить к входу IN2 компаратора сигнал, формируемый ЦАП2. Определить дифференциальную и общую нелинейность преобразования путем анализа результатов аналого-цифрового преобразования, сохраняемых во внутреннем ОЗУ с адреса 20H. Выход из программы – кнопкой «Сброс».

Проверить линейность и стабильность «быстрого» преобразования, заменив команду с меткой VV на CALL SADC.

4) Исследовать следящий метод аналого-цифрового преобразования. В качестве имитатора сигнала использовать ЦАП2. Определить максимально возможную скорость изменения измеряемого сигнала. Смену выходного уровня ЦАП2 синхронизировать прерыванием от таймера 0 управляющей программой:

```
        ORG     4000H
        JMP     START
        ORG     400BH
        MOV     32H,#1
        JMP     TMSAU       ;обслуживание прерывания таймера 0
START:  CALL    INITT0
        MOV     DPH,#80H    ;адрес ЦАП2
        CLR A
SS:     MOVX    @DPTR,A
        CALL    SLADC
         SETB F0
        ADD     A,#16
M:      JB      F0,M        ;сброс бита F0 производит TMSAU
        JMP SS
```

Отчет по лабораторной работе должен содержать:

- Тексты программ с комментариями.
- Осциллограммы переходных процессов.
- Результаты оценки нелинейности АЦП.

Контрольные вопросы для самопроверки

1. Что представляет собой аналого-цифровое преобразование?
2. Чем характеризуются точность и линейность преобразования?
3. Какие функции выполняет устройство выборки хранения?
4. Как осуществляется преобразование методом поразрядного уравновешивания?

5. В чём заключается следящий метод преобразования?
6. Какие недостатки имеет следящий метод по сравнению с методом поразрядного уравновешивания?
7. Какими техническими средствами лабораторной установки реализуется аналого-цифровое преобразование?
8. Как осуществляется на макете преобразование методом поразрядного уравновешивания?
9. Как осуществляется на макете преобразование следящим методом?

# Лабораторная работа 9
# Программирование прерываний микроконтроллера 87C51FA

**Цель работы:** исследование и программирование системы прерываний микроконтроллера

## Теоретические сведения

### Система прерываний

Использование системы прерываний позволяет микроЭВМ реагировать на некоторые внешние и внутренние события практически в момент их возникновения. Задержка в зависимости от типа выполняемой текущей команды составляет от 3 до 7 машинных циклов. МикроЭВМ заканчивает выполнение текущей команды, идентифицирует источник и вызывает подпрограмму обработки прерывания, аппаратно генерируя команду CALL. Подпрограммы обработки прерывания располагаются в ПЗУ микроЭВМ по фиксированным адресам (векторам).

| | |
|---|---|
| INT0 | 0003H |
| Timer0 | 000BH |
| INT1 | 0013H |
| Timer1 | 001BH |
| Serial Port | 0023H |
| Timer2 | 002BH |
| PCA | 0033H |

На каждую подпрограмму обработки прерывания отведено всего 8 байт. Этого объема может не хватить для размещения программы, поэтому по указанным адресам чаще всего располагают безусловный переход на подпрограмму, которая может находиться в произвольной области ПЗУ.

Источник прерывания может быть внешний (входы INT0, INT1) и внутренний (таймеры, массив программируемых счетчиков (PCA), последовательный порт и др.). При обнаружении прерывания устанавливается соответствующий флаг. Последовательность опроса флагов и, соответственно, обслуживания прерываний с одинаковым уровнем приоритета такова:

1. IE0 – флаг прерывания по входу INT0.
2. TF0 – флаг переполнения таймера 0.
3. IE1 – флаг прерывания по входу INT1.
4. TF1 – флаг переполнения таймера.
5. CF, CCF*n* – флаги массива программируемых счетчиков, *n* = 0 ... 4.
6. TI, RI – флаги передатчика и приемника последовательного порта.
7. TF2, EXF2 – флаги переполнения таймера 2.

Если источник прерывания имеет более одного флага, установка в единицу любого из них приводит к вызову одной и той же подпрограммы обработки (логика «ИЛИ»), которая сама должна распознать причину прерывания.

Управление системой прерываний производится путем установки/сброса бит в регистре разрешения прерываний IE и двух регистрах приоритетов IP и IPH. Каждый из источников прерывания может быть индивидуально разрешен или запрещен установкой или сбросом соответствующего бита в регистре IE. Сброс в «0» бита EA запрещает сразу все прерывания.

IE.7 IE.0

| EA | EC | ET2 | ES | ET1 | EX1 | ET0 | EX0 |
|---|---|---|---|---|---|---|---|
| Enable | PCA | Timer2 | Serial | Timer1 | INT1 | Timer0 | INT0 |

Установка/сброс разрядов в младшем (IP) и старшем (IPH) регистрах приоритетов позволяет назначить источнику прерывания один из четырех уровней приоритета: 00 – низший, 01, 10, 11 – высший.

Адрес 0B8H (IP)

IP.7 IP.0

| – | PPC | PT2 | PS | PT1 | PX1 | PT0 | PX0 |
|---|---|---|---|---|---|---|---|
| Резерв | PCA | Timer2 | Serial | Timer1 | INT1 | Timer0 | INT0 |

Адрес 0B7H (IPH)

| – | PPCH | PT2H | PSH | PT1H | PX1H | PT0H | PX0H |
|---|---|---|---|---|---|---|---|
| Резерв | PCA | Timer2 | Serial | Timer1 | INT1 | Timer0 | INT0 |

Регистры IE и IP допускают битовую адресацию, т. е. возможны команды типа CLR EA, SETB PT1 и т. п. В регистр IPH информацию можно записать только байтом, например MOV 0B7H,#08H.

Вызов подпрограммы обработки блокирует прерывания, имеющие такой же уровень приоритета, как и обрабатываемый источник, т. е. источники одного уровня не прерывают обработку друг друга. Источник с более высоким уровнем приоритета может прервать текущую подпрограмму обработки (вложенные прерывания). При обнаружении запросов на прерывание одновременно двух или более источников первыми обрабатываются имеющие более высокий уровень приоритета, а внутри одного уровня – в соответствии с последовательностью опроса флагов. Опрос флагов производится в каждом машинном цикле в пятом такте.

Любое прерывание может быть вызвано программно путем установки флага командой SETB.

Настройка на тип сигнала прерывания по входам INT0, INT1 производится битами IT0, IT1, которые расположены в регистре TCON, имеющем битовую адресацию.

Адрес 088H (TCON)

TCON.7 TCON.0

| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|---|---|---|---|---|---|---|---|

TF*n* – флаг переполнения таймера;

TR*n* – бит пуска (1), останова (0) таймера;

IE*n* – флаг прерывания по входу INT*n*;

IT*n* – бит управления типом прерывания («1» – по срезу, «0» – уровнем);

(*n* = 0, 1).

При IT*n* = 0 уровень сигнала на входе INT*n* копируется в флаг IE*n* в пятом такте машинного цикла. Поэтому длительность нулевого уровня сигнала источника прерывания на входе микроЭВМ должна быть не менее одного машинного цикла.

Любой флаг прерывания может быть программно сброшен. Сброс флагов источников T0, T1 и INT0, INT1 при работе по перепаду на входе производится аппаратно при переходе на подпрограмму обработки прерывания. При работе по уровню флаги прерывания по входам INT*n* должны быть программно сброшены в соответствующих подпрограммах обработки. Флаги прерываний PCA, таймера 2 и последовательного порта сбрасываются только программно.

Подпрограмма обработки прерывания должна оканчиваться командой RETI, которая восстанавливает содержимое счетчика команд (возврат в основную программу) и снимает блокировку прерываний с уровнем приоритета обслуженного источника.

По сигналу RESET регистры IE, IP, IPH, TCON обнуляются.

## Задание

1) Ознакомиться с описанием.

2) Разместить по адресам 4003H, 400BH и 4013H подпрограммы обработки прерываний от входа INT0, таймера 0 и INT1 в соответствии с индивидуальным заданием.

*Примечание. В лабораторном макете программа «монитор» производит переадресацию векторов прерываний в область адресов 4000H.*

3) Составить и отладить программу, использующую прерывания, проверить результаты ее работы на лабораторном макете.

В ходе работы управлять битами разрешения прерываний отдельных источников, соединяя входы T0, T1 с шиной «Общий». В соответствии с заданием зарисовать осциллограммы или описать характер вывода информации на дисплей.

В качестве сигнала прерывания по INT0 и INT1 использовать частоту, вырабатываемую генератором макета или вызывать прерывания, подключая соответствующий вход к "Общему".

Назначить всем источникам прерываний низкий приоритет. Запретить прерывания всех источников, кроме INT0 и INT1. Далее:

а) запрограммировать прерывания от INT0 и INT1 по перепаду. Разрешать прерывания: поочередно для INT0 и INT1;

б) запрограммировать прерывания от INT0 по уровню, от INT1 по перепаду. Разрешить прерывания совместно для INT0 и INT1;

в) назначить высокий приоритет INT1. Разрешить прерывания совместно для INT0 и INT1;

г) разрешить только прерывания таймера 0. Анализировать результат работы управляющей программы, изменяя период прерывания таймера (перед запуском программы вводить директивой монитора в ячейку 32H числа 10H, 80H, F0H). Разрешить прерывания INT0 и INT1 совместно с прерываниями таймера 0. Назначить таймеру высший приоритет.

В ходе выполнения работы анализировать осциллограммы на выходе ЦАП1 и управляемых линий порта P1.

Управляющая программа:

```
        ORG     4000H
        JMP     START
        ORG     4003H
        JMP     NX0         ;на п/п обработки прерывания INT0
        ORG     400BH
        JMP     NT0         ;на п/п обработки прерывания таймера 0
        ORG     4013H
        JMP     NX1         ;на п/п обработки прерывания INT1
NT0:    PUSH    ACC
        MOV     A,32H       ;ввод из ячейки 32H коэффициента
        CPL     A           ;k, задающего период счета (прерывания)
        INC     A           ;k·250 мкс – период таймера 0
        MOV     TH0,A       ;занесение коэффициента в регистр
        POP     ACC         ;таймера
        ***             ;команды, реализующие индивидуальное задание
        RETI
NX0:
        ***             ;команды, реализующие индивидуальное задание
        RETI
```

```
NX1:
        ***                 ;команды, реализующие индивидуальное задание
        RETI
START:
        CALL    INITT0      ;настройка таймера 0
        MOV     P1,#16
;настройка регистров системы прерываний
        MOV     0B7H,#00000010B         ;высший приоритет T0
        SETB    PX1   ;(CLR PX1) оставить требуемую по заданию команду
        SETB    ET0   ;(CLR ET0) оставить требуемую по заданию команду
M:      MOV     C,T0        ;вход порта P3.4
        MOV     EX0,C       ;разрешение/запрет прерывания INT0
        MOV     C,T1        ;вход порта P3.5
        MOV     EX1,C       ;разрешение/запрет прерывания INT1
        JMP     M
        END
```

**Варианты индивидуальных заданий**

1. В подпрограмме прерывания INT0 инвертировать состояние линии P1.4. По прерыванию INT1 инвертировать состояние линии P1.5. По прерыванию от таймера 0 инкрементировать содержимое порта P1.

2. Сформировать на ЦАП1 напряжение, нарастающее на 1 дискрет при обработке прерывания по входу INT0. При обработке прерывания по входу INT1 сформировать на ЦАП1 напряжение, убывающее на 2 дискрета. По прерыванию от таймера 0 установить флаг прерывания IE0, вызвав прерывание от INT0 программно (в этом случае на вход INT0 сигнал не подавать).

3. Сформировать на ЦАП1 напряжение, нарастающее на 3 дискрета при обработке прерывания по входу INT0. При обработке прерывания по входу INT1 сформировать на ЦАП1 напряжение, убывающее на 1 дискрет. По прерыванию от таймера 0 установить флаг прерывания IE1, вызвав прерывание от INT1 программно (в этом случае на вход INT1 сигнал не подавать).

4. Сформировать на ЦАП1 напряжение, нарастающее на 4 дискрета при обработке прерывания по входу INT0. Обнулить содержимое регистра P1 при обработке прерывания по входу INT1. По прерыванию от таймера 0 установить флаг прерывания IE1, вызвав прерывание от INT1 программно (в этом случае на вход INT1 сигнал не подавать).

5. По прерыванию INT0 декрементировать содержимое регистра P1. По прерыванию INT1 инкрементировать содержимое регистра P1. По прерыванию таймера 0 инвертировать содержимое бита T0.

6. По прерыванию INT0 устанавливать в «1» бит P1.7. По прерыванию INT1 сбрасывать в «0» бит P1.7. По прерыванию таймера 0 увеличивать содержимое регистра P1 на 4 дискрета.

7. По прерыванию INT0 производить циклический сдвиг регистра P1 вправо. По прерыванию INT1 производить циклический сдвиг регистра P1 влево. По прерыванию таймера 0 инвертировать содержимое бита T0.

8. По прерываниям INT0 и INT1 производить суммирование «по модулю два» содержимого регистра P1 с числом F0H. По прерыванию таймера 0 производить декремент содержимого регистра P1.

Отчет по лабораторной работе должен содержать:

- Вариант задания.
- Тексты программ с комментариями.
- Осциллограммы на выходе ЦАП1 и управляемых линиях порта P1.

Контрольные вопросы для самопроверки

1. Что такое прерывание?
2. Как программируются прерывания?
3. Как определяется приоритет прерываний?
4. В каких регистрах размещаются маски и флаги прерываний?
5. Чем отличаются подпрограммы обслуживания прерываний от обычных?
6. Как инициировать прерывания программным путём?
7. Сколько уровней приоритета прерываний можно запрограммировать?
8. Как сбрасываются флаги прерываний?
9. Что называют вложенными прерываниями?

# Лабораторная работа 10

# Программирование таймеров микроконтроллера 87C51FA

**Цель работы:** Изучение и программирование таймеров-счётчиков.

### Теоретические сведения

В состав периферийных устройств микроконтроллера имеет входят три аппаратных таймера/счетчика. Каждый имеет два восьмиразрядных регистра TH*n* (старший) и TL*n* (младший), *n* = 0, 1 и 2. Установка соответствующих бит управления позволяет использовать устройство как счетчик внешних событий или как формирователь интервалов времени (таймер).

В режиме таймера регистр TL*n* инкрементируется в каждом машинном цикле. В лабораторном макете установлен кварц с частотой $F_{КВ}$ = 12МГц. Длительность машинного цикла равна $12/F_{КВ}$, поэтому инкремент таймера происходит с периодом 1 мкс.

В режиме счетчика внешних событий регистр TL*n* инкрементируется при обнаружении перепада 1/0 на входном контакте таймера (P3.4 для таймера 0, P3.5 для таймера 1, P1.0 для таймера 2). Анализ состояния входной линии производится в пятом такте каждого машинного цикла, и если в двух соседних циклах обнаружены состояния «1» и «0», происходит аппаратный инкремент младшего регистра счетчика. Таким образом, максимальная частота, которую способен подсчитывать счетчик, составляет $F_{КВ}/24$. Длительность высокого и низкого уровней на входе должна быть не меньше одного машинного цикла.

Управление работой таймеров 0 и 1 осуществляется путем записи информации в регистр режимов TMOD (нет битовой адресации) и регистр управления TCON (есть битовая адресация), которые по сигналу RESET обнулены.

Адрес 089H (TMOD)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| GATE1 | C/T1 | M1.1 | M0.1 | GATE0 | C/T0 | M1.0 | M0.0 |

GATE*n* = 0 – запрещено управление от внешнего вывода INT*n*;
GATE*n* = 1 – разрешено управление от внешнего вывода INT*n*;
C/T*n* = 0 – режим таймера;
C/T*n* = 1 – счетчик внешних событий;
M1.*n*, M0.*n* – двоичный код номера режима;
(*n* = 0, 1).

При наличии «1» в бите GATE*n* работа соответствующего таймера/счетчика возможна только при высоком уровне сигнала на входе INT*n*. Подача на этот вход низкого уровня остановит инкремент, в регистрах таймера/счетчика будет сохранено последнее значение кода. Управление от внешнего вывода позволяет использовать таймер в качестве измерителя длительности интервалов времени.

Адрес 088H (TCON)

TCON.7 TCON.0

| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|---|---|---|---|---|---|---|---|

TF*n* – флаг переполнения таймера;

TR*n* – бит пуска/останова таймера;

IE*n* – флаг прерывания по входу INT*n*;

IT*n* – управление типом прерывания («1» – по перепаду, «0» – уровнем).

В регистре TCON расположены биты управления таймерами и входами внешних прерываний. Установка бита TR*n* в «1» разрешает счет, сброс – переводит таймер в режим хранения.

Переполнение таймера/счетчика (переход «все единицы» – «все нули») аппаратно устанавливает флаг TF*n*, формируя запрос прерывания. Сброс флага – программный или аппаратный (при вызове подпрограммы обработки прерывания). Регистры TH*n* и TL*n* доступны как для записи, так и для чтения.

**Режимы работы таймера 0 и таймера 1.**

***Режим 0.*** M1 = 0, M0 = 0. 13-разрядный таймер/счетчик, использует восемь разрядов TH и пять младших разрядов TL. Режим исторически восходит к таймеру семейства MCS48, используется редко.

***Режим 1.*** M1 = 0, M0 = 1. 16 разрядный таймер/счетчик, используются восемь разрядов TH и восемь разрядов TL.

В режимах 0 и 1 установка TF происходит при переполнении регистра TH.

***Режим 2.*** (автозагрузка) M1 = 1, M0 = 0. Восьмиразрядный таймер/счетчик на основе регистра TL. При переполнении TL устанавливается флаг переполнения TF и происходит аппаратная запись в TL содержимого TH, который выступает в качестве регистра хранения коэффициента счета. Запись в TH – программная.

***Режим 3.*** M1 = 1, M0 = 1. Таймер 0 представляет собой два независимых устройства на основе восьмиразрядных регистров TH0 и TL0. Устройство TL0 использует все биты управления таймера 0 и устанавливает флаг TF0 при переполнении регистра TL0. Устройство TH0 работает только в режиме таймера (вход – $F_{КВ}/12$), устанавливает флаг TF1 при переполнении регистра TH0, пуск/останов устройства TH0 производится битом TR1.

При работе таймера 0 в режиме 3 таймер 1 может быть запрограммирован в любой из первых трех режимов, однако при этом он не будет устанавливать флаг прерывания и лишен управления битом TR1 (остановка возможна только внешним сигналом низкого уровня на входе INT1 при GATE1 = 1). При использовании таймера 1 в режиме 2 для тактирования последовательного порта флаг прерывания не нужен. При установке таймера 1 в режим 3 он блокируется и сохраняет содержимое своих регистров.

Остановка таймеров сбросом бит TR*n* или низким уровнем на входах INT*n* не изменяет содержимого регистров TH*n*, TL*n*. Во всех режимах, кроме режима 2, после переполнения счет начинается с кода «все нули», что соответствует максимальному коэффициенту счета. Программная запись чисел в регистры TH*n*, TL*n* меняет коэффициент счета. В режиме 2 коэффициент счета хранится в регистре TH и производится аппаратная запись содержимого TH в регистр TL после его переполнения.

Использование аппаратных таймеров 0 или 1 предполагает следующие действия:

– выбор источника сигнала, режима работы и способа управления;

– занесение соответствующего управляющего слова в регистр TMOD;

– занесение в регистры TH*n*, TL*n* кода коэффициента счета;

– пуск таймера установкой бита TR*n* в единицу.

Результаты работы таймера обрабатываются программно путем использования прерываний либо с помощью опроса флагов TF*n*.

**Программирование и режимы работы таймера 2**

Во всех режимах работы таймер 2 представляет собой 16 разрядный счетчик TH2, TL2, который для хранения кодов использует восьмиразрядные регистры RCAP2H, RCAP2L. Выбор режима работы и способа управления таймером 2 осуществляется путем установки/сброса бит в регистре управления T2CON (есть битовая адресация) и записи управляющего слова в регистр режима T2MOD (нет битовой адресации). По сигналу RESET оба регистра обнуляются.

Адрес 0C8H (T2CON)

| T2CON.7 | | | | | | | T2CON.0 |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2 | CP/RL2 |

TF2 – бит переполнения. Устанавливается в «1» аппаратно при переполнении регистра TH2, вызывает прерывание по адресу 002BH. Сброс только программный. При RCLK = 1 или TCLK = 1 аппаратная установка TF2 в единицу блокируется.

EXF2 – флаг внешнего управляющего входа T2EX (контакт порта P1.1). При EXEN2 = 1 флаг устанавливается в «1» при обнаружении на входе T2EX перепада 1/0. Вызывает прерывание по адресу 002BH. Сброс только программный.

RCLK и TCLK – биты подключения приемника и передатчика последовательного порта. При сбросе бита последовательный порт использует для тактирования соответствующего устройства таймер 1, при установке в «1» – таймер 2.

EXEN2 – флаг разрешения внешнего управляющего входа T2EX. Сброс EXEN2 в «0» отключает T2EX от управления таймером 2.

TR2 – пуск/останов таймера 2. Установка в «1» разрешает счет.

C/T2 = 0 – устройство в режиме таймера, вход – $F_{КВ}$ /12.

C/T2 = 1 – устройство в режиме счетчика перепадов 1/0 на входе T2 (P1.0).

CP/RL2 = 0 – режим автозагрузки.

CP/RL2 = 1 – режим защелкивания.

Регистр T2MOD содержит два управляющих бита, используемых только в режиме автозагрузки. При работе таймера 2 в двух других режимах эти разряды не программируются и содержат «0».

T2MOD Адрес 0C9H

| – | – | – | – | – | – | T2OE | DCEN |
|---|---|---|---|---|---|---|---|

«–» – резервные биты, запись «1» запрещена.

T2OE – бит разрешения формирования меандра на выводе P1.0.

DCEN – бит разрешения управления направлением (+/-) счета TH2, TL2.

***Режим защелкивания***

CP/RL2 = 1, RCLK = TCLK = 0. Управление TR2, C/T2.

Бит EXEN2 = 0. Счетчик/таймер инкрементирует код в TH2, TL2 и при переполнении устанавливает TF2 в единицу.

Бит EXEN2 = 1. Текущее состояние TH2, TL2 записывается в регистры RCAP2H, RCAP2L в момент обнаружения перепада 1/0 на входе T2EX. Одновременно устанавливается EXF2. Счетчик/таймер при этом продолжает счет и при переполнении установит TF2 в единицу.

Режим защелкивания может использоваться для измерения периода сигнала, подаваемого на вход P1.1 (T2EX). Для этого необходимо вычислить разность двух последовательно «защелкнутых» двухбайтных кодов. Результат выражен в количестве машинных циклов (при $F_{КВ}$ = 12МГц – в микросекундах).

***Режим автозагрузки***

а) CP/RL2 = 0, RCLK = TCLK = 0, T2OE = 0, DCEN = 0.

Управление TR2, C/T2.

В регистры RCAP2H, RCAP2L программно записывают дополнительный 16 разрядный код коэффициента счета. Счетчик/таймер инкрементирует TH2, TL2 и при переполнении аппаратно переписывает в TH2, TL2 содержимое RCAP2H, RCAP2L, восстанавливая коэффициент счета. Одновременно устанавливается в единицу TF2.

Если бит EXEN2 = 1, автозагрузка произойдет также и при обнаружении перепада 1/0 на входе T2EX, одновременно будет установлен в единицу флаг EXF2.

б) CP/RL2 = 0, RCLK = TCLK = 0, T2OE = 0, DCEN = 1.

Управление TR2, C/T2.

Режим с управлением направлением счета. Бит EXEN2 игнорируется.

Если на входе T2EX «1», счетчик TH2, TL2 инкрементируется и при достижении кода FFFFH производится аппаратная перезапись содержимого RCAP2H, RCAP2L в TH2, TL2, установка TF2 в «1», инверсия содержимого EXF2 (аналогично действию команды CPL).

Если на входе T2EX «0», счетчик TH2, TL2 декрементируется и при совпадении его кода с содержимым RCAP2H, RCAP2L производится аппаратная запись в TH2, TL2 кода FFFFH, установка TF2 в единицу и инверсия содержимого EXF2.

Единичный уровень бита EXF2 не вызывает прерывание, его содержимое может быть использовано как 17-й разряд реверсивного счетчика, направление счета которого управляется входом T2EX.

в) CP/RL2 = 0, RCLK = TCLK = X, T2OE = 1, DCEN = X, C/T2 = 0.

X – безразличное состояние. Управление TR2.

Генератор частоты. На выводе P1.0 по переполнению TH2, TL2 формируется меандр с частотой $F_{КВ}$ / [4(65536 - RCAP2H, RCAP2L)]. Таймер не воздействует при переполнении на флаг TF2. Если бит EXEN2 = 1, перепад 1/0 на входе T2EX устанавливает EXF2 и вызывает прерывание, но не влияет на формирование таймером частоты.

***Режим тактирования последовательного порта***

CP/RL2 = X, RCLK или TCLK = 1, T2OE = X, DCEN = X, C/T2 = 0.

Управление TR2.

По переполнению таймера формируется тактирующий сигнал для последовательного порта с частотой $F_{КВ}$ / [32(65536 - RCAP2H,RCAP2L)].

Флаг TF2 не задействован. Перепад 1/0 на входе T2EX вызывает установку EXF2 и прерывание, если EXEN2 = 1, но не влияет на работу таймера. Для обмена со скоростью 9600 бит/с при $F_{КВ}$ = 12МГц следует записать в RCAP2H код 0FFH, в RCAP2L код 0D9H.

Использование таймера 2 предполагает следующие действия:

– выбор режима работы, источника входного сигнала, способа управления и установку соответствующих бит в регистре T2CON;

– в режиме автозагрузки – запись управляющего слова в T2MOD;

– занесение в регистры RCAP2H, RCAP2L числа, определяющего коэффициент счета (дополнительный код);

– пуск таймера установкой бита TR2 в единицу.

***ВНИМАНИЕ!*** *По сигналу RESET (нажатие кнопки «Сброс») в лабораторном макете запускается программа «монитор», которая программирует таймер 2 в режим тактирования последовательного порта со скоростью 9600 бит/с.*

## Задание

1) Ознакомиться с теоретической частью работы.

2) Исследовать работу таймера 0 в периодическом режиме.

Запрограммировать его в режим таймера с автозагрузкой (режим 2). Запустить программу формирования меандра с периодом 100 мкс по прерываниям таймера 0 на выводе P1.7.

3) Исследовать работу таймера 2 в режиме автозагрузки

а) Запрограммировать таймер 2 в режим генератора меандра, занести в RCAP2L код 255, в RCAP2H код 244. Отладить и запустить циклическую программу формирования 10 периодов линейно меняющейся частоты на выводе T2. Для этого в программе выполнить инкремент кода в RCAP2H при обнаружении перепада 1/0 на выводе T2. Рассчитать крайние значения формируемых частот.

б) Запрограммировать таймер 2 в режим счетчика с автозагрузкой и управлением по входу T2EX. Коэффициент счета 16. На вход T2 (P1.0) подать счетные импульсы – меандр 100 мкс с выхода P1.7 (программа п. 2). Запустить циклическую программу, копирующую содержимое TL2 в ЦАП2 один раз за период меандра. Подать на вход T2EX сигнал с генератора макета. Синхронизировать осциллограф по сигналу генератора макета. Изменяя частоту генератора, зарисовать характерные осциллограммы на выходе ЦАП2.

4) Исследовать работу таймера 2 в режиме защелкивания. Измерить период меандра, формируемого на выводе P1.7 (программа п.2). Измерить максимальный и минимальный периоды сигнала, формируемого генератором макета. В циклической программе измерения разность кодов двух последовательных защелкиваний сохранить в ячейках ОЗУ 30H (младший байт) и 31H. Для вывода содержимого 31H, 30H на экран в десятичном коде вызвать подпрограмму OUTMEM.

5) Реализовать на микроЭВМ «счетчик времени». Для этого таймер 1 запрограммировать с периодом счета 1 с в режиме счетчика внешних импульсов. В качестве внешнего сигнала для таймера 1 использовать меандр с периодом 100 мкс на выводе P1.7 (программа п. 2). Написать программу вывода раз в секунду на экран текущего времени в формате YY.XX (Y – минуты, X – секунды). Инкремент кода времени и вывод производить по переполнению таймера 1.

Отчет по лабораторной работе должен содержать:

- Тексты программ с комментариями.
- Осциллограммы сигналов.

Контрольные вопросы для самопроверки

1. Какие таймеры/счётчики входят в состав устройств микроконтроллера 87C51FA?
2. В каких режимах могут работать таймеры/счётчики 0 и 1?
3. Что представляет собой режим автозагрузки у таймеров/счётчиков?
4. Для чего нужен бит GATE?
5. Перечислите режимы и функции таймера/счётчика 2.
6. Что происходит в режиме защёлкивания и как он программируется?
7. Как запрограммировать таймер/счётчик 2 на формирование меандра?
8. Как реализуется функция ШИМ таймера/счётчика 2?
9. Как реализуется режим тактирования последовательного порта таймером/счётчиком 2?
10. Как реализуется на таймере/счётчике 2 режим часов?

# Лабораторная работа 11

## Работа с массивом программируемых счётчиков микроконтроллера 87C51FA

**Цель работы:** исследование режимов работы и программирование PCA

### Теоретические сведения

Массив программируемых счетчиков (Programmable Counter Array – PCA) обеспечивает более широкие возможности формирования и измерения временных интервалов с меньшим отвле-

чением центрального процессора, чем стандартный таймер/счетчик. К преимуществам PCA следует отнести более высокую точность, которую он может обеспечить по сравнению с таймерами. Некоторые задачи, например, измерение фазового сдвига между двумя сигналами или формирование широтно-импульсно-модулированного (ШИМ) сигнала, решаемые с помощью PCA, невозможно решить стандартными таймерами без привлечения дополнительных аппаратно-программных средств.

PCA состоит из 16 разрядного таймера/счетчика и пяти 16 разрядных модулей сравнения/защелки. Таймер PCA является базовым таймером для всех модулей, ни один другой таймер микроЭВМ не может быть использован для этой цели.

Текущий код таймера PCA хранится в восьмиразрядных регистрах CH, CL, доступных для записи и чтения в любой момент времени. По сигналу источника тактовых импульсов регистр CL инкрементируется.

Программирование таймера PCA производится путем записи управляющего слова в регистр режима CMOD (нет битовой адресации) и установки бит в регистре управления CCON (есть битовая адресация). По сигналу RESET используемые таймером биты обнуляются.

Адрес 0D9H (CMOD)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CIDL | WDTE | – | – | – | CPS1 | CPS0 | ECF |

ECF – бит разрешения прерывания. При установке в «1» переполнение таймера PCA вызовет прерывание.

CPS1, CPS0 – код источника тактовых импульсов таймера:

00 – $F_{КВ}$/12, инкремент CL в такте 5 каждого машинного цикла;

01 – $F_{КВ}$/4, инкремент CL в тактах 1, 3, 5 каждого машинного цикла;

10 – таймер 0, инкремент CL в машинном цикле переполнения таймера 0;

11 – внешний сигнал, инкремент CL при обнаружении перепада 1/0 на входе ECI (P1.2). Анализ состояния входа производится в тактах 1, 3, 5 каждого машинного цикла, таким образом, частота входного сигнала не должна превышать $F_{КВ}$/8.

Регистр CH инкрементируется после переполнения CL в следующем такте машинного цикла.

WDTE = 1 – разрешает работу сторожевого таймера.

Бит CIDL анализируется при переходе в режим пониженного энергопотребления:

CIDL = 1 – таймер останавливается;

CIDL = 0 – счет не прекращается.

Адрес 0CD8H (CCON)

CCON.7 CCON.0

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| CF | CR | – | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |

CF – флаг прерывания, устанавливается аппаратно при переполнении таймера PCA. Сброс только программный. При ECF = 1 установка CF в «1» вызовет прерывание по адресу 0033H.

CR – пуск/останов таймера. Сброс/установка программная, установка в «1» разрешает счет.

CCF*n* (*n* = 0 ... 4) – флаг прерывания модуля с номером *n*. Устанавливается аппаратно. Сброс только программный. При наличии «1» в бите ECCF*n* установка CCF*n* в «1» вызовет прерывание по адресу 0033H.

Следует отметить, что все шесть источников прерывания массива программируемых счетчиков имеют один общий вектор прерывания. Прерывание произойдет, если в регистре IE установлены биты EA и EC и произошла установка в «1» хотя бы одного из разрешенных битами ECF, ECCF*n* флагов прерывания. Подпрограмма обработки прерывания, расположенная по адресу 0033H, должна определить конкретный источник прерывания, опросив флаги CF, CCF0 …

CCF4, и выполнить требуемые источником действия. Последняя команда должна сбросить флаг обрабатываемого источника.

Модули сравнения/защелки. Каждый из пяти модулей (*n* = 0 ...4) имеет пару восьмиразрядных регистров CCAP*n*H, CCAP*n*L, доступных для записи/чтения. Для взаимодействия с внешней средой модули используют контакты порта P1, которым присвоены символические имена CEX*n* (CEX0 – P1.3, CEX1 – P1.4, CEX2 – P1.5, CEX3 – P1.6, CEX4 – P1.7). Программирование режима работы модуля производится путем записи управляющего слова в соответствующий регистр CCAPM*n* (не имеет битовой адресации). По сигналу RESET все регистры CCAPM*n* обнуляются.

CCAPM*n*

| — | ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF*n* |
|---|---|---|---|---|---|---|---|

ECOM – установка в «1» разрешает модулю функцию сравнения.

CAPP – установка в «1» разрешает защелкивание по перепаду 0/1.

CAPN – установка в «1» разрешает защелкивание по перепаду 1/0.

MAT – запись «1» разрешает модулю устанавливать флаг CCF*n* в режимах программного таймера и высокоскоростного выхода.

TOG = 1 включает режим высокоскоростного выхода.

PWM = 1 включает режим широтно-импульсной модуляции.

ECCF*n* – бит разрешения прерывания. Запись «0» блокирует прерывание модуля при установке флага CCF*n*. Каждый модуль сравнения/защелки может работать в любом из пяти режимов.

**Режим защелки:**

***– по перепаду 0/1*** (CCAPM*n* = XX10.000XB). X – безразличное состояние.

Модуль производит анализ состояния входа CEX*n* в тактах 1, 3, 5 машинного цикла. При обнаружении перепада 0/1 происходит аппаратная загрузка содержимого регистров таймера CH, CL в регистры модуля CCAP*n*H, CCAP*n*L. Устанавливается в «1» флаг CCF*n*. Таймер счет не останавливает. Следующий перепад 0/1 вызовет запись в регистры новых значений. Старые будут утеряны.

Для измерения временного интервала между перепадами необходимо сохранить в ОЗУ предыдущее значение «защелкнутого» 16 разрядного кода. На это при использовании прерывания уходит не менее 9 машинных циклов. Следовательно, перепады не должны следовать чаще, чем один раз в десять машинных циклов. Вычитая из последнего «защелкнутого» значения предыдущее, получим время между двумя перепадами в числе периодов источника тактовых импульсов таймера PCA. В лабораторном макете при подключении таймера к $F_{КВ}$/12 результат будет в микросекундах;

***– по перепаду 1/0*** (CCAPM*n* = XX01.000XB).

Аналогичен предыдущему. Отличается направлением перепада, аппаратно обнаруживаемым на входе CEX*n*;

***– по любому перепаду*** (CCAPM*n* = XX11.000XB).

Содержимое CH, CL загружается в CCAP*n*H, CCAP*n*L при обнаружении перепада 0/1 или 1/0 на входе CEX*n*. Режим может использоваться для измерения длительности импульса (единичного или нулевого уровня).

***ВНИМАНИЕ!*** *В режиме защелки в бит порта P1, служащий входом, необходимо занести «1» для обеспечения альтернативной функции порта.*

**Режим высокоскоростного выхода** (CCAPM*n* = X100.110XB).

Предназначен для выработки сигнала на выводе CEX*n* через заданный промежуток времени. Модуль использует функцию сравнения. При совпадении кодов регистров CCAP*n*H, CCAP*n*L с содержимым CH, CL устанавливается флаг CCF*n* и происходит инверсия уровня сигнала на выходе CEX*n*.

Для формирования сигналов с заданным периодом после обнаружения очередного совпадения кодов (обычно по прерыванию CCF*n*) необходимо модифицировать CCAP*n*H, CCAP*n*L, добавив к их содержимому код, вычисленный как отношение времени между двумя сигналами на выходе CEX*n* к периоду тактовых импульсов таймера PCA. При записи числа в CCAP*n*L бит ECOM*n* автоматически сбрасывается, при записи в CCAP*n*H – автоматически устанавливается в «1», снова разрешая функцию сравнения. Указанная последовательность записи должна соблюдаться программой пользователя. Аппаратный сброс/установка ECOM*n* при модификации регистров модуля PCA исключает ложный вызов прерывания в момент, когда новое содержимое CCAP*n*H, CCAP*n*L еще не записано окончательно.

После записи нового значения кода в CCAP*n*H, CCAP*n*L необходимо сбросить CCF*n* и, если требуется, программно записать в CEX*n* исходное состояние. При установке CEX*n* в «1» на выходе с заданным периодом формируются импульсы с активным нулевым уровнем, при сбросе в «0» – с единичным. Отсутствие записи в CEX*n* исходного уровня приводит к формированию меандра с удвоенным периодом.

Если не модифицировать содержимое CCAP*n*H, CCAP*n*L, очередное совпадение произойдет через $2^{16}$ тактов таймера PCA. В этом случае в макете при тактовой частоте $F_{КВ}/12$ минимальная частота меандра на выходе CEX*n* составляет 7.63 Гц, при $F_{КВ}/4$ – 22.9 Гц. Подпрограммы с именами F7 и F23 формируют сигналы указанных частот в режиме высокоскоростного выхода на выводе P1.7 (модуль 4).

**Режим программного таймера** (CCAPM*n* = X100.100XB).

Аналогичен режиму высокоскоростного выхода. Режим не использует контакт порта P1. Результатом работы является установка бита прерывания CCF*n* при совпадении кодов CH, CL и CCAP*n*H, CCAP*n*L.

**Режим ШИМ** (CCAPM*n* = X100.0010B).

Используется автозагрузка восьмиразрядного кода, хранящегося в CCAP*n*H, в регистр CCAP*n*L. Модуль PCA производит сравнение содержимого регистров CCAP*n*L и CL. Если CCAP*n*L< CL, на выходе CEX*n* установлен сигнал нулевого уровня, если CCAP*n*L ≥ CL, на выходе CEX*n* сигнал единичного уровня. Период ШИМ-сигнала на выходе CEX*n* равен периоду переполнения регистра CL и зависит от частоты тактовых импульсов таймера PCA.

Код, записанный в CCAP*n*L, задает скважность формируемого сигнала. При переполнении регистра CL производится аппаратная загрузка содержимого CCAP*n*H в CCAP*n*L. Программная загрузка кода в CCAP*n*L может привести к сбою при формировании очередного импульса, поэтому следует модулировать длительность загружая код в CCAP*n*H. Коду 00H соответствует максимальная длительность (постоянный единичный уровень), коду 80H – половина периода (меандр), коду FFH – импульс минимальной длительности (1/256 периода). Постоянный нулевой уровень можно получить только программным сбросом бита CEX*n* в «0».

ШИМ-сигнал может быть использован для преобразования цифровых кодов в аналоговый сигнал с минимальными аппаратными затратами (RC-цепь в качестве низкочастотного фильтра).

Режим не использует флаг CCF*n*. Для обеспечения альтернативной функции порта при работе модуля PCA в режиме ШИМ необходимо установить соответствующий бит P1 (CEX*n*) в «1».

В лабораторном макете при подключении таймера к $F_{КВ}/12$ период ШИМ составляет 256 мкс (3.9 кГц), при использовании $F_{КВ}/4$ – 85.3 мкс (11.8 кГц). Подпрограммы с именами T256 и T85 формируют на выводе P1.6 меандр с указанными периодами, используя модуль 3 в режиме ШИМ.

**Режим сторожевого таймера** (CCAPM4 = X100.1X0XB).

Функцию сторожевого таймера может выполнять только модуль 4. Режим обеспечивает аппаратный сброс (RESET) микроЭВМ, если к сторожевому таймеру не поступит регулярный сигнал, подтверждающий нормальное функционирование системы, в которую включена микроЭВМ.

В момент совпадения содержимого регистров CH, CL и CCAP4H, CCAP4L генерируется внутренний сигнал сброса. Для того чтобы сброс не произошел, необходимо периодически изменять значение в регистрах CCAP4H, CCAP4L таким образом, чтобы вновь записанное значение до новой записи не могло совпасть с CH, CL.

Внешний контакт порта CEX4 и флаг CCF4 сторожевой таймер не использует.

## Задание

В качестве источника тактовых импульсов для таймера PCA использовать частоту $F_{КВ}/12$.

1) Запрограммировать модуль 0 в режим защелкивания по одному из перепадов. Отладить и запустить циклическую программу измерения периода сигнала, поданного на вход CEX0. Результатом измерения является разность двух последовательных «защелкиваний» CH, CL в регистрах CCAP0H, CCAP0L. Сохранить результат в ячейках 30H (младший байт) и 31H. Для вывода его на экран вызвать подпрограммы OUTMEM и DSEC.

На вход CEX0 (P1.3) подать сигнал с генератора макета. Изменяя частоту, определить максимальное и минимальное значения периода формируемого сигнала.

2) Разрешить в модуле 0 «защелкивание» по обоим перепадам (1/0 и 0/1). Измерить длительность уровня «1» сигнала генератора. Записать максимальное и минимальное значения. Измерить длительность уровня «0» сигнала.

3) Запрограммировать модули 1, 2 и 3 в режим высокоскоростного выхода. Сформировать на выходах меандры с периодом 2050, 2048 и 2052 мкс соответственно. Синхронизировать осциллограф по сигналу первого модуля (2050 мкс). Наблюдать сигналы на выходах 1 и 2, 1 и 3.

4) Сформировать на базе PCA систему программной «подстройки» частоты. Для этого модуль 0 запрограммировать в режим защелкивания по перепаду 1/0. Модуль 1 запрограммировать в режим высокоскоростного выхода. По сигналу прерывания модуля 0 вычислять период частоты на входе CEX0, по сигналу прерывания модуля 1 прибавлять к содержимому CCAP1H, CCAP1L половину вычисленного периода (для формирования меандра переключение выхода CEX1 должно происходить дважды за период). Запустить циклическую программу. Подать на вход CEX0 сигнал с генератора макета. Изменяя частоту генератора, наблюдать сигналы на контактах CEX0 и CEX1. При отключении выхода генератора от CEX0 на выходе CEX1 должна сохраняться «подстроенная» частота.

5) Запрограммировать в режим ШИМ модуль, выход которого подключен к ФНЧ. Управлять длительностью импульса по закону модулированного синуса, последовательно загружая в регистр CCAP2H коды из массива TABMOD. Выводить те же коды в ЦАП2, используя подпрограмму DAC2 (выводит содержимое аккумулятора). Выход CEX2 (P1.5) подключен к входу ФНЧ. Сравнить аналоговые сигналы на выходах ЦАП2 и ФНЧ.

Запрограммировать таймер PCA на использование в качестве источника тактовых импульсов $F_{КВ}/4$. Запустить программу управления ШИМ. Сравнить результаты работы.

*Указание: для синхронизации момента вывода модулирующих кодов в CCAP2H и ЦАП2 с частотой ШИМ-сигнала можно воспользоваться меандром, формируемым на выводе CEX3 (P1.6) модулем 3 в режиме ШИМ (содержимое CCAP3H = 128). Запуск меандра – вызов подпрограммы T85 для $F_{КВ}/4$ и подпрограммы T256 для $F_{КВ}/12$.*

6) Для источников тактовых импульсов $F_{КВ}/12$ и $F_{КВ}/4$ рассчитать максимальный период, в течение которого необходимо в выполняемой программе перезагрузить содержимое регистров сторожевого таймера макета, чтобы не произошел перезапуск программы. Предложить способы формирования этого интервала времени.

Отчет по лабораторной работе должен содержать:

- Тексты программ и измеренные длительности сигналов.
- Объяснение причин возникновения погрешности при измерении периода и длительностей с помощью PCA.
- Комментарии характера осциллограмм по пп. 3 – 5.

Контрольные вопросы для самопроверки

1. Какие устройства входят в состав PCA?
2. Какие функции могут выполнять модули PCA?
3. Как реализуется режим защёлки?
4. Как работает PCA в режиме высокоскоростного выхода?
5. Что представляет собой режим программного таймера?
6. Как реализуется режим ШИМ?
7. Как программируется сторожевой таймер?
8. Для чего используется режим высокоскоростного выхода?
9. Какой вид имеют сигналы на выходе ШИМ?

# Лабораторная работа 12

## Программирование памяти микроконтроллера C8051F064 в среде разработки SiLabs IDE

**Цель работы:** Ознакомление с устройством аппаратной поддержки разработок микропроцессорных систем и приобретение навыков программирования оперативной памяти микроконтроллера C8051F064

### Теоретические сведения

Есть две области памяти микроконтроллера C8051F064: флэш-память программ и оперативная память данных, каждая из которых имеет своё адресное пространство. В памяти программ хранятся программный код и константы. В памяти данных – оперативные данные. Размер ячейки памяти программ два байта, ячейки памяти данных – один байт. Память данных может иметь до 64 килобайт адресов, из которых 256 адресов принадлежат внутренней памяти, а остальные внешней.

В пространстве внутренней оперативной памяти находятся регистры общего назначения, регистры специальных функций, ячейки с побитной адресацией и ячейки внутренней памяти, относящиеся к сверхоперативной памяти микроконтроллера. В частности, в этом пространстве находится адрес регистра указателя вершины стека – области памяти, в которой сохраняются во время выполнения подпрограмм содержимое счётчика команд и других регистров, например, регистра слова состояния микроконтроллера.

На примере работы со стеком можно ознакомиться с методикой программирования оперативной памяти. Стек размещается в любом выбранном месте 256-байтной памяти данных. Выбор области размещения стека выполняется с помощью указателя стека (Stack Pointer SP,0x81).

**ПРИМЕР 1** – работа со стеком и оперативной памятью

```
 mov SP, #0x10 ;инициализация стека
mov 0x21, #15h   ;инициализация памяти по адресам 21-22
 mov 0x22, #18h
push 0x21 :запись в стек из ячейки памяти
pop 0x71 ;чтение из стека в ячейку памяти
```

Создание проекта на ассемблере

1. Создаём рабочую папку для проекта.
2. Добавляем в папку файл (c8051F060.inc) определения регистров.
3. Запускаем интегрированную среду разработки (IDE.exe).
4. Сохраняем проект в рабочую папку (Project -> Save Project As...).
5. Создаём основной файл (File -> New File).
6. Сохраняем файл в рабочую папку под именем "main.asm" (File -> Save As…).
7. Добавляем main.asm в проект (Add Files to Project, кликая правой кнопкой по значку проекта в левой панели).
8. Добавляем main.asm в сборку (кликая правой кнопкой по значку файла).
9. Проект создан. После, необходимо сохранить проект.

Создание проекта на языке программирования Си.

1. Создаём рабочую папку для проекта.
2. Добавляем в папку файл (c8051F060.h) определения регистров.
3. Запускаем интегрированную среду разработки (IDE) .
4. Сохраняем проект в рабочую папку (Project -> Save Project As...).
5. Создаём основной файл (File -> New File).
6. Сохраняем файл в рабочую папку под именем "main.c" (File -> Save As...).
7. Добавляем main.c в проект (Add Files to Project, кликая правой кнопкой по значку проекта в левой панели).
8. Добавляем main.c в сборку (кликая правой кнопкой по значку файла)
9. Запускаем мастер конфигурации (Config2).
10. Выбираем семейство и модель используемого микроконтроллера (C8051F064).
11. Сохраняем проект конфигурации в рабочую папку (File -> Save Project As).
12. Сохраняем файл исходного кода конфигурации в рабочую папку под именем "config.c".
13. Переходим в IDE, добавляем config.c в проект и в сборку (аналогично пунктам 6 и 7).

Проект создан. После совершения изменений в мастере конфигурации необходимо сохранить проект конфигурации (File -> Save Project) и файл исходного кода конфигурации config.c (Save Source As...)

Аналогично программируются другие регистры и ячейки памяти. Так как в микроконтроллере используются разные виды запоминающих устройств, то целесообразно указывать в программе сегмент памяти.

**ПРИМЕР 2 –** создание проекта

```
$include (c8051F060.inc); Подключение файла определения регистров;
CSEG AT 00h; выбор сегмента.
ljmp Main; переход на начало программы.
cseg at 1500h; сегмент памяти начинается с адреса 1500.
Main: mov WDTCN, #0DEh ; Отключение сторожевого таймера.
      mov WDTCN, #0ADh ; Отключение сторожевого таймера.
      mov SP, #0x15; инициализация стека с 15.
      ; инициализация памяти по адресам 50h to 53h.
      mov 0x50, #24h; запись числа 24 по адресу 50h
      mov 0x51, #F2h; запись числа F2 по адресу 51h
      mov 0x52, #D3h; запись числа D3 по адресу 52h
```

```
        mov 0x53, #14h; запись числа 14 по адресу 53h
        ; запись в стек по адресам 50h to 53h с увеличением значения стека на единицу.
        push 0x50; запись в стек по адресу 50h и увеличения значения на единицу.
        push 0x51; запись в стек по адресу 50h и увеличения значения на единицу.
        push 0x52; запись в стек по адресу 50h и увеличения значения на единицу.
        push 0x53; запись в стек по адресу 50h и увеличения значения на единицу.
        ; чтение из стека и копирование по адресам 50h to 53h.
        pop 0x50; чтение из стека и копирование в адрес 50h.
        pop 0x51; чтение из стека и копирование в адрес 51h.
        pop 0x52; чтение из стека и копирование в адрес 52h.
        pop 0x53; чтение из стека и копирование в адрес 53h.
        nop; команда холостого хода
end
```

## Задание

1. Создать проект в среде SILABS IDE на ассемблере или Си.
2. Инициализировать стек.
3. Инициализировать память по адресу согласно варианту.
4. Произвести запись в стек с увеличением на единицу и скопировать содержимое в память данных.
5. Выполнить чтение из стека и скопировать содержимое в память данных.
6. Проверить работу программы.

Таблица 12.1 Варианты

| Вариант | Адреса памяти | стек | данные |
|---|---|---|---|
| 1. | 21,22,23,24 | 12 | 12,F2,D3,24 |
| 2. | 41,42,43,44 | 13 | F2,12,D3,24 |
| 3. | 51,52,53,54 | 14 | 24,D3,F2,12 |
| 4. | 61,62,63,64 | 15 | F2,24,12,D3 |
| 5. | 71,72,73,74 | 16 | 12,D3,F2,24 |
| 6. | 81,82,83,74 | 17 | D3,F2,23,24 |
| 7. | 91,92,93,94 | 18 | 21,22,23,24 |
| 8. | 75,77,78,79 | 19 | 21,22,23,24 |
| 9. | 32,33,34,35 | 20 | 21,22,23,24 |
| 10. | 56,57,58,59 | 21 | 21,22,23,24 |

Отчет по лабораторной работе должен содержать:

- Схему алгоритма
- Листинг программы
- Выводы

Контрольные вопросы для самопроверки

1. Какие виды запоминающих устройств реализованы в микроконтроллере?
2. Какой размер у адресного пространства оперативной памяти?
3. Каков объем внутренней оперативной памяти и что в неё входит?
4. Что такое стек?
5. Как инициализируется стек?
6. Как создать проект на ассемблере?
7. Как создать проект на языке Си?
8. Как определяется сегмент программы?
9. Как данные заносятся в стек и извлекаются из стека?

# Лабораторная работа 13

### Программирование параллельных портов ввода-вывода микроконтроллера C8051F064 в среде разработки SiLabs IDE

**Цель работы:** исследование работы портов ввода/вывода восьмиразрядного микроконтроллера C8051F064.

### Теоретические сведения

Микроконтроллер C8051F064 имеет восемь 8-битных портов, разделённых на две группы. Первая группа портов ввода/вывода P0-P3(младшая группа) использует побайтовую и побитовую адресацию, выполняющуюся с помощью регистров специального назначения. Вторая группа портов ввода/вывода P4-P7 (старшая группа) имеет только побайтовую адресацию. Все порты конфигурируются по двухтактной (Push-Pull) схеме или схеме с открытым стоком (Open-Drain).

Старшая группа может применяться в качестве портов ввода/вывода или интерфейса внешней памяти. Выводы младшей группы портов применяют как входы/выходы цифровых периферийных устройств.

На отладочной плате кнопка S3, соединена с выводом порта P3.6 микроконтроллера, а светодиод D3, соединён с выводом порта P1.6. Резистор R7 предназначен для ограничения тока через светодиод, R3 - для подтяжки вывода микроконтроллера к напряжению питания при отпущенной кнопке, R4 - для ограничения тока через вывод микроконтроллера в случае, если кнопка будет нажата в то время, как порт будет настроен на вывод, C23 предотвращает "дребезг" кнопки (когда одно нажатие воспринимается как несколько).

Для коммутации выводов микроконтроллера, в структуру ввода/вывода включается цифровая матрица (Crossbar), регулируемая приоритетным дешифратором, состояние которого, устанавливается с помощью управляющих регистров: XBR0, XBR1, XBR2 (Port I/O register) и P1MDIN (Port 1 Input Mode Register).

Светодиод загорается в том случае, если линия порта P1.6 сконфигурирована как push-pull и выводится логическая единица. При отпущенной кнопке на вывод P3.6 микроконтроллера, через резистор R3, поступает напряжение питания, которое соответствует логической единице, нажатая кнопка соединяет этот вывод с общим проводом, что соответствует логическому нулю.

**ПРИМЕРЫ**

**Программа управления светодиодом на языке Си**

```
#include "c8051F060.h" // Подключение файла определения регистров;
sbit BUTTON = P3^6; // Присвоить имя BUTTON кнопке, соединенной с выводом порта P3.6;
sbit LED = P1^6;  // Присвоить имя LED выводу порта P1.6;
void Init_Device(); // Обращение к подпрограмме;
void main() {
        Init_Device(); // Подпрограмма;
        LED = 0; // Вывести логический ноль в порт P1.6
        while(1) {; // Цикл
                while(BUTTON == 1);          // Задержка до нажатия кнопки;
                while(BUTTON == 0);          // Задержка до отпускания кнопки;
                LED = 1; // Вывести логическую единицу в порт P1.6
                while(BUTTON == 1); // Задержка до нажатия кнопки;
                while(BUTTON == 0); // Задержка до отпускания кнопки;
                LED = 0; // Вывести логический ноль в порт P1.6
        }
}
```

**Программа управления светодиодом на языке Ассемблер**

```
$include (C8051F060.inc); Подключение файла определения регистров;
LED equ P1.6 ; Присвоить имя BUTTON кнопке соединенной с выводом порта P3.6;
Button equ P3.6 ; Присвоить имя LED выводу порта P1.6
 cseg AT 0 сегмент памяти начинается с адреса 00
ljmp main ; Переход к подпрограмме;
main:
mov   WDTCN, #0DEh ; Отключение сторожевого таймера.
mov   WDTCN, #0ADh ; Отключение сторожевого таймера.
mov     SFRPAGE,#CONFIG_Page ; Выбор страницы SFR
 mov   XBR2, #40h  ; Включить коммутатор
orl   P1MDOUT,#40h ; Переключить порт P1.6 в режим push-pull
clr   LED ; Задание исходного состояния светодиода
metka:
jb Button, metka ; Ожидание нажатия кнопки
cpl LED ; Вывести логическую единицу в порт P1.6 (загорается светодиод)
lcall DEL ; Переход к подпрограмме
DEL:
mov R7,#03h ; Организация задержки
clr LED ; Вывести логический ноль в порт P1.6 (отключается светодиод)
mov R7,#03h ; Организация задержки
ljmp metka ; Зацикливание программы
end
```

## Задание

1. Создать проект в среде SILABS IDE (СМ лабораторную работу №1).
2. Открыть мастер конфигурации(config2)

3. В мастере конфигурации открыть диалог настройки источников сброса (Peripherals -> Reset Sources), перейти на вкладку сторожевого таймера (Watchdog Timer), отключить его (флажок Disable Watchdog Timer). Если этого не сделать, микроконтроллер будет постоянно совершать сброс через небольшие промежутки времени.
4. Открыть диалог настройки портов ввода-вывода (Peripherals -> Port I/O), включить коммутатор (Enable Crossbar) и переключить порт P1.6 в двухтактный режим (push-pull).
5. Написать программу для управления светодиодом при помощи кнопки согласно варианту на указанном преподавателем языке программирования.
6. Проверить работу программы.

Таблица 13.1 Варианты

| Вариант | Задание |
|---|---|
| 1. | Светодиод загорается сразу после сброса (до нажатия кнопки). |
| 2. | Светодиод загорается после однократного нажатия и отпускания кнопки, а гаснет после двукратного нажатия и отпускания. |
| 3. | Светодиод загорается после двукратного нажатия и отпускания кнопки, а гаснет после однократного нажатия и отпускания. |
| 4. | Светодиод загорается и гаснет в момент нажатия кнопки. |
| 5. | Светодиод загорается в момент нажатия кнопки, а при повторном нажатии гаснет в момент отпускания. |
| 6. | Светодиод загорается в момент отпускании кнопки, а при повторном нажатии гаснет в момент нажатия. |
| 7. | Светодиод загорается только на время нажатия кнопки, а при отпускании сразу гаснет. |
| 8. | Светодиод загорается после двукратного нажатия и отпускания кнопки, а гаснет после её двукратного нажатия и отпускания. |
| 9. | Светодиод загорается в момент отпускании кнопки, а при повторном двукратном нажатии гаснет в момент второго нажатия. |
| 10. | Светодиод загорается при отпускании кнопки и гаснет после повторного нажатия и отпускания в момент отпускания. |

Отчет по лабораторной работе должен содержать:

- Цель работы
- Задание
- Блок-схему алгоритма
- Листинг программы
- Выводы

Контрольные вопросы для самопроверки

1. Сколько параллельных портов у микроконтроллера C8051F064?
2. Какие порты можно адресовать побитно?
3. Как могут конфигурироваться линии портов?
4. Как настроить линию порта на вывод на ассемблере?
5. Как настроить линию порта на вывод на Си?
6. Какие средства проверки работы портов есть на плате C8051F064EK?

7. Как реализовать управление светодиодом D3 на плате C8051F064EK кнопкой S3?
8. Какие альтернативные функции выполняют линии параллельных портов?
9. Какие функции выполняет коммутатор Crossbar?
10. Каково назначение мастера конфигурации?

# Лабораторная работа 14

## Программирование таймеров и внешних прерываний микроконтроллера C8051F064 в среде разработки SiLabs IDE

**Цель работы:** исследование таймеров и системы прерываний, связанной с их работой микроконтроллера C8051F064.

### Теоретические сведения

Микроконтроллер C8051F064 содержит 5 таймеров, два из которых имеют совместимы с таймерами/счетчиками МК семейства 8051, а три других 16-битные таймеры с автоперезагрузкой, применяемые вместе с аналого-цифровыми преобразователями, контроллерами последовательных интерфейсов или в качестве таймеров общего назначения. Все таймеры имеют два 8-битных счётных регистра, старший(High) и младший(Low): TH0, TL0, TH1, TL1.

Работа таймера регулируется тремя регистрами: CKCON (Clock Control Register) регистр синхронизации, TMOD (Timer Mode Register) регистр режима и TCON (Timer Control Register) регистр управления.

Таймеры 0 и 1

Режим работы 0

13-битный таймер/Счетчик. В этом режиме регистр таймера TH0 разделен на 8 бит, расположенных в старшем регистре и 5 бит, расположенных в младшем регистре.

Режим работы 1

16-битный таймер/счетчик, в этом режиме работа таймеров осуществляется схоже с режимом 0, но отличается тем, что регистры TL0 и TL1 переполняются при значении FFFFH.

Режим работы 2

Регистр TL0 используется для счета, а регистр TH0 для хранения перезагружаемого значения. При переполнении регистра TL0 устанавливается флаг переполнения TF0 и происходит перезагрузка из регистра TH0.

Режим работы 3

в этом режиме регистры TH0 и TL0 работают как два отдельных 8-битных таймера/счетчика.

Таблица 14.1 Регистр управления TCON

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| Значения после сброса: | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 |

Бит 7 TF1- флаг переполнения таймера 1.
Бит 6 TR1-бит управления таймера 1.
Бит 5 TF0-флаг переполнения таймера 0.
Бит 4 TR0-бит управления таймера 0.

Бит 3 IE1- флаг внешнего прерывания 1.
Бит 2 IT1- бит выбора типа внешнего прерывания 1.
Бит 1 IE0-флаг внешнего прерывания 0.
Бит 0 IT0- маска внешнего прерывания 0.

Таблица 14.2 Регистр режима TMOD

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| GATE1 | C/T1 | T1M1 | T1M0 | GATE0 | C/T0 | T0M1 | T0M0 |
| Значения после сброса: | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 |

Бит 7 GATE1-бит управления стробом таймера 1.
Бит 6 C/T1-бит выбора режима таймер/счетчик.
Бит 5,4 T1M1, T1M0-биты выбора режимов таймера 1.
Бит 3 GATE0- бит управления стробом таймера 0.
Бит 2 C/T0- бит выбора режима таймер/счетчик.
Бит 1,0 T0M1, T0M0- биты выбора режимов таймера 0.
Таймер 2

Это 16-битный таймер/счётчик со счётным регистром TH2:TL2. Может считать импульсы синхронизации (режим таймера) или перепады сигнала на внешнем выходе T2 (режим счётчика).

Режим работы 0

Таймер/счетчик с захватом. При поступлении активного сигнала на вход захвата содержимое счётного регистра переносится в регистр захвата.

Режим работы 1

Таймер/счетчик с автоперезагрузкой, При переполнении таймера 2 устанавливается флаг TF2, генерируется прерывание и содержимое регистра перезагрузки загружается в счётный регистр, после чего, таймер продолжает свою работу.

Режим работы 2

В этом режиме таймер 2 используется для синхронизации последовательного интерфейса UART0, когда он работает в режимах 3 или 1.

Таблица 14.3 Регистр управленияT2CON

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK0 | TCLK0 | EXEN2 | TR2 | C/T2 | CP/RL2 |
| | Значения после сброса: | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 |

Бит 7 TF2 - флаг переполнения таймера 2.
Бит 6 EXF2 - внешний флаг таймера 2.
Бит 5 RCLK0 -флаг синхронизации приемника UART0.
Бит 4 TCLK0 - флаг синхронизации передатчика UART0.
Бит 3 EXEN2 - разрешение внешнего входа таймера 2.
Бит 2 TR2 - бит управления таймера 2.
Бит 1 C/T2 - бит выбора режима таймер/счетчик.
Бит 0 CP/RL2 - бит выбора режима захват/перезагрузка.

Таймер 3

16-битный таймер со счётным регистром TMR3H:TMR3L. Может работать только в режиме автоперезагрузки. При синхронизации от внешнего генератора используется как часы реального времени.

Таблица 14.4 Регистр управления TMR3CN

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| TF3 | - | - | - | - | TR3 | T3M | T3XCLK |
| Значения после сброса: | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 |

Бит 7 TF3 – флаг переполнения таймера 3.
Бит 6-3 не используются.
Бит 2 TR3 – бит управления таймера 3.
Бит 1 T3M – выбор синхронизации таймера 3.
Бит 0 T3XCLK – выбор внешней синхронизации таймера 3.

Таймер 4

16-битный таймер/счётчик со счётным регистром TH4:TL4. Имеет три основных режима работы: 16-битный таймер/счетчик с захватом, 16-битный таймер/счетчик с автоперезагрузкой, генератор импульсов синхронизации для последовательного интерфейса UART1.

Режим 0

16-битный таймер/счетчик с захватом устанавливается с помощью бита выбора захват/перезагрузки CP/RL4, разрешения входа таймера 4 EXEN4 (Timer 4 External Enable) и бита управляющего запуском таймера 4.

Режим 1

16-битный таймер/счетчик с автоперезагрузкой, в котором переполнение таймера приводит к установке флага TF4, соответствующему прерыванию и перезагрузке счётного регистра.

Режим 2

генератор импульсов синхронизации для последовательного интерфейса в этом режиме таймер 4 используется для синхронизации интерфейса UART1, когда он работает в режимах 3 или 1.

Таблица 14.5 Регистр управления T4CON

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| TF4 | EXF4 | RCLK1 | TCLK1 | EXEN4 | TR4 | C/T4 | CP/RL4 |
| Значения после сброса: | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит 1 | Бит 0 |

Бит 7 TF4 – флаг переполнения таймера 4.
Бит 6 EXF4 – внешний флаг таймера 4.
Бит 5 RCLK1 –флаг синхронизации приемника UART1.
Бит 4 TCLK1 – флаг синхронизации передатчика UART1.
Бит 3 EXEN4 – разрешение внешнего входа таймера 4.
Бит 2 TR4 – бит управления таймера 4.
Бит 1 C/T4 – бит выбора режима таймер/счетчик.
Бит 0 CP/RL4 – бит выбора режима захват/перезагрузка.

**ПРИМЕР**

```
#include «c8051F060.h» // подключение файла определения регистров;
sbit BUTTON = P3^6 // присвоить имя BUTTON кнопке, соединенной с выводом порта P3.6;
sbit LED = P1^6  // присвоить имя LED выводу порта P1.6;
void Init_Device()
void main() {
        Init_Device()
        while(1) {
                while(BUTTON) //ожидание пока будет нажата кнопка
                while(!BUTTON) //ожидание отпускания кнопки
                TCON = 0x00; //инициализация таймера
                while(BUTTON) //ожидание пока будет нажата кнопка
                while(!BUTTON) //ожидание пока будет нажата кнопка
                TCON = 0x10 //запуск таймера
        }
}
void timer0_interrupt() interrupt 1 { //организация прерывания
        LED = ~LED;// светодиод начинает мигать
}
```

## Задание

1. Создать пустой проект
2.Открыть мастер конфигурации (config2)
3. Отключить сторожевой таймер
4. Переключить порт светодиода в режим push-pull (аналогично п. 3. Лаб. Раб. 1)
5. В мастере конфигурации открыть диалог настройки таймеров (Peripherals -> Timers), включить таймер 0 (Enable Timer на вкладке Timer 0), перевести его в 16-битный режим (16 Bit Counter/Timer) и задать в качестве источника тактовых импульсов SYSCLK / 48.
6. Открыть диалог настройки прерываний (Peripherals -> Interrupts или кнопка Configure Timer Interrupts в диалоге настройки таймеров), разрешить глобальные прерывания (Enable All Interrupts) и разрешить прерывание таймера 0 (Enable Timer 0 Interrupt).
7. Написать программу варианта задания, осуществляющую мигание светодиодом по прерыванию таймера с заданной частотой на языке Си или ассемблере (по указанию преподавателя)

Таблица 14.6 Варианты заданий

| вариант | задание |
|---|---|
| 1 | Получить аналогичный результат, используя таймер 1 вместо таймера 0 (приоритет прерывания 3) |
| 2 | Изменив настройки делителя тактовой частоты, увеличить частоту мигания светодиода. |
| 3 | Изменив разрядность счётчика, увеличить частоту мигания таймера 0 в 8 раз. |
| 4 | Выполнить задание 1 на таймере 2. |
| 5 | Выполнить задание 2 на таймере 2. |

Отчет по лабораторной работе должен содержать:

- Цель работы
- Задание
- Блок-схему алгоритма?

- Листинг программы
- Выводы

Контрольные вопросы для самопроверки

- В каких режимах могут работать таймеры 0 и 1?
- Какие режимы и функции реализуются на базе таймера 2?
- Как реализуется режим часов реального времени в таймере 3?
- Какие функции может выполнять таймер 4?
- Что представляет собой режим захвата?
- Как реализуется автоперезагрузка счётного регистра?
- По каким событиям генерируются прерывания в таймерах/счётчиках?
- В каких режимах генерируются импульсы синхронизации последовательного интерфейса?
- Какие регистры обеспечивают работу таймеров/счётчиков?
- Как программируется инициализация таймера/счётчика?

# Лабораторная работа 15

### Работа с универсальным асинхронным приёмопередатчиком UART микроконтроллера C8051F064 в среде разработки SiLabs IDE

**Цель работы:** исследование методов программирования последовательного интерфейса UART микроконтроллера C8051F064 в среде IDE.

### Теоретические сведения

В состав микроконтроллера C8051F064 входят два последовательных интерфейса UART: UART0 и UART1, в использующие одинаковый набор регистров, битов управления и флагов. UART (Universal Asynchronous Receiver-Transmitter) или универсальный асинхронный приёмопередатчик (УАПП) является улучшенным последовательным портом с нахождение ошибок (формата) и аппаратным обнаружением ведомого.

УАПП может работать в четырех режимах:

- Режим 0 8-битный синхронный режим с фиксированной скоростью
- Режим 1 8-битный асинхронный режим с изменяющейся скоростью
- Режим 2 8-битный асинхронный режим с фиксированной скоростью
- Режим 3 8-битный асинхронный режим с изменяющейся скоростью

Режим 0

В этом режиме контроллер обеспечивает синхронную полудуплексную связь. Приём и передача данных(DATA) осуществляются по линии RX, а для импульсов синхронизации CLK используется линия TX. Передача данных начинается в момент записи данных в буферный регистр SBUF.

Режим 1

8-разрядный UART с асинхронной связью и регулируемой скоростью передачи данных. Посылка содержат 10 бит: один стартовый бит (STAR BIT), 8-бит данных (D0-D7) и 1 бит остановки (STOP BIT). Данные передаются на вывод TX и принимаются с вывода RX. 8 бит данных принимаются с линии в буферный регистр SBUF, а стоп-бит записывается в ячейку RB8 регистра управления SCON.

Режим 2

9-разрядный UART с асинхронной связью и фиксированной скоростью передачи данных. Посылка состоит из 11 бит: стартового бита, 8 бит данных, программируемого девятого бита данных и стоп-бита. Режим 2 поддерживает организацию связи с несколькими микроконтроллерами (мультипроцессорное взаимодействие). При передаче девятый бит данных выбирается из ячейки TB8, а при сохраняется в ячейке RB8 регистра управления SCON.

Режим 3

9- разрядный UART с асинхронной связью и изменяемой скоростью передачи данных. По протоколу передачи данных, аналогичен режиму 2 а генерация скорости передачи данных, такая же как режиме 1. В режиме 3 передаются 11 бит: 1 бит старта, 8 бит данных, программируемый девятый бит данных и один стоп-бит. Скорость передачи данных задаётся таймерами 1, 2, 3 или 4. Поддерживается мультипроцессорный режим.

Таблица 15.1 Регистр управления SCON0

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
|---|---|---|---|---|---|---|---|
| SM00/FE0 | SM10/RX0V0 | SM20/TXC0L0 | REN0 | TB80 | RB80 | TI0 | RI0 |
| Значения после сброса: | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Бит 7 | Бит 6 | Бит 5 | Бит 4 | Бит 3 | Бит 2 | Бит1 | Бит 0 |

Бит 7, 6 Функция этих битов определяется состоянием бита SSTAT0 (PCON.6) и, в зависимости от его состояния, или соответствуют флагам ошибок, или задают режим работы UART.
Бит 5 SM20 – разрешение мультипроцессорной связи.
Бит 4 REN0 – разрешение приема.
Бит 3 TB80 – девятый передаваемый бит.
Бит 2 RB80 - девятый принимаемый бит.
Бит 1 TI0 – флаг прерывания передатчика.
Бит 0 RI0 - флаг прерывания приемника.

**ПРИМЕР**

```
#include "c8051F060.h" // Подключение файла определения регистров;
#include <stdio.h> // Подключение файла, где объявляются функции;
void Init_Device();
char uart_receive() {
        while(RI0 == 0) // Ожидание сброса флага прерывания приемника;
        RI0 = 0 // Сброс флага прерывания приемника;
        return SBUF0; прием
}
void uart_transmit(char c) {
        while(TI0 == 0) // Ожидание сброса флага прерывания передатчика;
        TI0 = 0 // Сброс флага прерывания передатчика;
        SBUF0 = c; //Передача
}
void main() {
        char c; // Определение локальной переменной
        Init_Device();
        TI0 = 1 // Установка флага прерывания передатчика;
        while(1) {
```

```
            printf("\nHello\n"); Вывести «Hello»
            c = uart_receive(); Обращение к подпрограмме
            printf("You typed: "); Вывести «You typed: »
            uart_transmit(c); Обращение к подпрограмме
        }
}
```

## Задание

1. Создать пустой проект
2.Открыть мастер конфигурации(config2)
2. Отключить сторожевой таймер.
3. Открыть диалог настройки портов ввода-вывода (Peripherals -> Port I/O), включить коммутатор (Enable Crossbar) и универсальный асинхронный приёмопередатчик (UART0)
4. Установить в качестве внешнего источника системной тактовой частоты кварцевый резонатор частотой 22.1184 МГц (в мастере конфигурации открыть диалог Peripherals -> Oscillators, вкладку External Oscillator, установить флажок Use External Oscillator as SYSCLK, выбрать верный диапазон частоты Frequency Control Bits и ввести частоту кварцевого резонатора в герцах)

5. В диалоге настройки последовательных интерфейсов (Peripherals -> UARTs) на вкладке UART0 задать режим 8 бит, изменяемой скорости передачи (8 Bit UART, Variable Baud Rate), включить приём (Enable UART0 Reception) и убедиться, что в качестве источника синхронизирующих импульсов для приёмника и передатчика установлен таймер 1

6. В диалоге настройки таймера 1 (Peripherals -> Timers, вкладка Timer 1) включить таймер (Enable Timer), установить 8-битный режим с автоматической перезагрузкой и задать коэффициент деления системной тактовой частоты (Clock Source и Prescaled Clock Inputs) и значение перезагрузки таймера, согласно рассчитанным для заданного варианта работы.

7. Написать программу на для приема через UART: «Hello» и передачи «You typed»

Таблица 15.2 Варианты заданий

| Вариант | Задание | | |
|---|---|---|---|
| | Скорость | SYSCLK | SYSCLK / 4 |
| 1 | 1200 | FE | B8 |
| 2 | 2400 | 70 | DC |
| 3 | 4800 | B8 | EE |
| 4 | 9600 | DC | F7 |
| 5 | 19200 | EE | FA |
| 6 | 38400 | F7 | FD |
| 7 | 57600 | FA | 70 |
| 8 | 115200 | FD | FE |

Отчет по лабораторной работе должен содержать:

- Цель работы

- Задание
- Блок-схему алгоритма?
- Листинг программы
- Выводы

Контрольные вопросы для самопроверки

- Сколько режимов рабаты у интерфейса UART?
- Как организуются передача и приём данных в режиме 0?
- Как передаются и принимаются данные в режиме 1?
- Для чего применяется и как организуется режим 3 работы UART?
- Чем отличается режим 3 от режима 2?
- Где хранится 9 бит при передаче и приёме?
- Для чего используют 9 бит?
- Как организуется мультипроцессорный режим?
- Как реализуется работа с UART в системе C8051F064EK?

## Приложение 1.

### СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА AT90S8515

Арифметические и логические операции

| Мнемоника | Операнды | Описание | Действия | Флаги | Циклы |
|---|---|---|---|---|---|
| ADD | Rd,Rr | Сложить два регистра | Rd = Rd + Rr | Z,C,N,V,H,S | 1 |
| ADC | Rd,Rr | Сложить два регистра с переносом | Rd = Rd + Rr + C | Z,C,N,V,H,S | 1 |
| SUB | Rd,Rr | Вычесть регистр | Rd = Rd - Rr | Z,C,N,V,H,S | 1 |
| SUBI | Rd,K8 | Вычесть константу | Rd = Rd - K8 | Z,C,N,V,H,S | 1 |
| SBC | Rd,Rr | Вычесть регистр с переносом | Rd = Rd - Rr - C | Z,C,N,V,H,S | 1 |
| SBCI | Rd,K8 | Вычесть константу 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 | Исключающее ИЛИ | Rd = Rd EOR Rr | Z,N,V,S | 1 |
| COM | Rd | Дополнение до 1 | Rd = $FF - Rd | Z,C,N,V,S | 1 |
| NEG | Rd | Дополнение до 2-х | Rd = $00 - Rd | Z,C,N,V,H,S | 1 |
| SBR | Rd,K8 | Установить биты в регистре | Rd = Rd V K8 | Z,C,N,V,S | 1 |
| CBR | Rd,K8 | Сбросить биты в регистре | Rd = Rd · ($FF - K8) | Z,C,N,V,S | 1 |
| INC | Rd | Увеличить регистр на 1 | Rd = Rd + 1 | Z,N,V,S | 1 |
| DEC | Rd | Уменьшить регистр на 1 | Rd = Rd -1 | Z,N,V,S | 1 |
| TST | Rd | Проверить на 0 или 1 | Rd = Rd · Rd | Z,C,N,V,S | 1 |
| CLR | Rd | Обнулить регистр | Rd = 0 | Z,C,N,V,S | 1 |
| SER | Rd | Установить регистр | Rd = $FF | – | 1 |

| | | | | | |
|---|---|---|---|---|---|
| ADIW | Rdl,K6 | Сложить слово с константой | Rdh:Rdl = Rdh:Rdl + K6 | Z,C,N,V,S | 2 |
| SBIW | Rdl,K6 | Вычесть из слова константу | Rdh:Rdl = Rdh:Rdl - K 6 | Z,C,N,V,S | 2 |
| MUL | Rd,Rr | Умножение целой части без знака | R1:R0 = Rd * Rr | Z,C | 2 |
| MULS | Rd,Rr | Умножение целой части со знаком | R1:R0 = Rd * Rr | Z,C | 2 |
| MULSU | Rd,Rr | Умножение целых частей (со знаком на беззнаковое) | R1:R0 = Rd * Rr | Z,C | 2 |
| FMUL | Rd,Rr | Умножение дробной части без знака | R1:R0 = (Rd * Rr) << 1 | Z,C | 2 |
| FMULS | Rd,Rr | Умножение дробной части со знаком | R1:R0 = (Rd *Rr) << 1 | Z,C | 2 |
| FMULSU | Rd,Rr | Умножение дробных частей (со знаком на беззнаковое) | R1:R0 = (Rd * Rr) << 1 | Z,C | 2 |

Таблица 2 Команды передачи управления

| **Мнемоника** | **Операнды** | **Описание** | **Действия** | **Флаги** | **Циклы** |
|---|---|---|---|---|---|
| RJMP | k | Относительный переход | PC = PC + k +1 | – | 2 |
| IJMP | – | Переход по адресу (Z) | PC = Z | – | 2 |
| EIJMP | – | Внешний переход по адресу (Z) | STACK = PC+1, PC(15:0) = Z, PC(21:16) = EIND | – | 2 |
| JMP | k | Переход | PC = k | – | 3 |
| RCALL | k | Относительный вызов под-программы | STACK = PC+1, PC = PC + k + 1 | – | 3/4* |
| ICALL | – | Вызов подпрограммы по ад-ресу (Z) | STACK = PC+1, PC = Z | – | 3/4* |
| EICALL | – | Extended Indirect Call to (Z) | STACK = PC+1, PC(15:0) = Z, PC(21:16) =EIND | – | 4* |
| CALL | k | Вызов подпрограммы | STACK = PC+2, PC = k | – | 4/5* |
| RET | – | Выход из подпрограммы | PC = STACK | – | 4/5* |
| RETI | – | Выход из подпрограммы пре-рывания | PC = STACK | I | 4/5* |
| CPSE | Rd,Rr | Сравнить, пропустить если равно | if (Rd ==Rr) PC = PC+2 or 3 | – | 1/2/3 |
| CP | Rd,Rr | Сравнение | Rd -Rr | Z,C,N,V,H,S | 1 |
| CPC | Rd,Rr | Сравнить с переносом | Rd - Rr - C | Z,C,N,V,H,S | 1 |
| CPI | Rd,K8 | Сравнить с константой | Rd - K | Z,C,N,V,H,S | 1 |
| SBRC | Rr,b | Пропустить, если бит реги-стра сброшен | if(Rr(b)==0) PC = PC + 2 or 3 | – | 1/2/3 |
| SBRS | Rr,b | Пропустить, если бит реги-стра установлен | if(Rr(b)==1) PC = PC + 2 or 3 | – | 1/2/3 |
| SBIC | P,b | Пропустить, если бит реги-стра ввода/вывода сброшен | if(I/O(P,b)==0) PC = PC + 2 or 3 | – | 1/2/3 |
| SBIS | P,b | Пропустить, если бит реги-стра ввода/вывода установлен | if(I/O(P,b)==1) PC = PC + 2 or 3 | – | 1/2/3 |
| BRBC | s,k | Переход, если флаг регистра статуса сброшен | if(SREG(s)==0) PC = PC + k + 1 | – | 1/2 |
| BRBS | s,k | Переход, если флаг регистра статуса установлен | if(SREG(s)==1) PC = PC + k + 1 | – | 1/2 |

| Мнемоника | Операнды | Описание | Действия | Флаги | Циклы |
|---|---|---|---|---|---|
| BREQ | k | Переход, если флаг нуля установлен | if(Z==1) PC = PC + k + 1 | – | 1/2 |
| BRNE | k | Переход, если флаг нуля сброшен | if(Z==0) PC = PC + k + 1 | – | 1/2 |
| BRCS | k | Переход, если установлен флаг переноса | if(C==1) PC = PC + k + 1 | – | 1/2 |
| BRCC | k | Переход, если перенос сброшен флаг переноса | if(C==0) PC = PC + k + 1 | – | 1/2 |
| BRSH | k | Переход, если равно или больше | if(C==0) PC = PC + k + 1 | – | 1/2 |
| BRLO | k | Переход, если меньше | if(C==1) PC = PC + k + 1 | – | 1/2 |
| BRMI | k | Переход, если минус | if(N==1) PC = PC + k + 1 | – | 1/2 |
| BRPL | k | Переход, если плюс | if(N==0) PC = PC + k + 1 | – | 1/2 |
| BRGE | k | Переход, если больше или равно со знаком | if(S==0) PC = PC + k + 1 | – | 1/2 |
| BRLT | k | Переход, если меньше со знаком | if(S==1) PC = PC + k + 1 | – | 1/2 |
| BRHS | k | Переход, если установлен флаг H | if(H==1) PC = PC + k + 1 | – | 1/2 |
| BRHC | k | Переход, если сброшен флаг H | if(H==0) PC = PC + k + 1 | – | 1/2 |
| BRTS | k | Переход, если установлен флаг T | if(T==1) PC = PC + k + 1 | – | 1/2 |
| BRTC | k | Переход, если сброшен флаг T | if(T==0) PC = PC + k + 1 | – | 1/2 |
| BRVS | k | Переход, если установлен флаг V | if(V==1) PC = PC + k + 1 | – | 1/2 |
| BRVC | k | Переход, если сброшен флаг V | if(V==0) PC = PC + k + 1 | – | 1/2 |
| BRIE | k | Переход, если разрешены прерывания | if(I==1) PC = PC + k + 1 | – | 1/2 |
| BRID | k | Переход, если прерывания запрещены | if(I==0) PC = PC + k + 1 | – | 1/2 |

Таблица 3. Команды пересылки данных

| Мнемоника | Операнды | Описание | Действия | Флаги | Циклы |
|---|---|---|---|---|---|
| MOV | Rd,Rr | Пересылка между регистрами | Rd = Rr | – | 1 |
| MOVW | Rd,Rr | Пересылка между парами регистров | Rd+1:Rd = Rr+1:Rr, r,d even | – | 1 |
| LDI | Rd,K8 | Загрузка константы в регистр | Rd = K | – | 1 |
| LDS | Rd,k | Прямая загрузка регистра | Rd = (k) | – | 2* |
| LD | Rd,X | Косвенная загрузка регистра | Rd = (X) | – | 2* |
| LD | Rd,X+ | Косвенная загрузка с постинкрементом | Rd = (X), X=X+1 | – | 2* |
| LD | Rd,-X | Косвенная загрузка с преддекрементом | X=X-1, Rd = (X) | – | 2* |
| LD | Rd,Y | Косвенная загрузка регистра | Rd = (Y) | – | 2* |

| | | | | | |
|---|---|---|---|---|---|
| LD | Rd,Y+ | Косвенная загрузка с постинкрементом | Rd = (Y), Y=Y+1 | – | 2* |
| LD | Rd,-Y | Косвенная загрузка с преддекрементом | Y=Y-1, Rd = (Y) | – | 2* |
| LDD | Rd,Y+q | Косвенная загрузка относительная | Rd = (Y+q) | – | 2* |
| LD | Rd,Z | Косвенная загрузка регистра | Rd = (Z) | – | 2* |
| LD | Rd,Z+ | Косвенная загрузка с постинкрементом | Rd = (Z), Z=Z+1 | – | 2* |
| LD | Rd,-Z | Косвенная загрузка с преддекрементом | Z=Z-1, Rd = (Z) | – | 2* |
| LDD | Rd,Z+q | Косвенная загрузка относительная | Rd = (Z+q) | – | 2* |
| STS | k,Rr | Прямое сохранение | (k) = Rr | – | 2* |
| ST | X,Rr | Косвенное сохранение | (X) = Rr | – | 2* |
| ST | X+,Rr | Косвенное сохранение с постинкрементом | (X) = Rr, X=X+1 | – | 2* |
| ST | -X,Rr | Косвенное сохранение с преддекрементом | X=X-1, (X)=Rr | – | 2* |
| ST | Y,Rr | Косвенное сохранение | (Y) = Rr | – | 2* |
| ST | Y+,Rr | Косвенное сохранение с постинкрементом | (Y) = Rr, Y=Y+1 | – | 2 |
| ST | -Y,Rr | Косвенное сохранение с преддекрементом | Y=Y-1, (Y) = Rr | – | 2 |
| ST | Y+ q,Rr | Косвенное сохранение относительное | (Y+q) = Rr | – | 2 |
| ST | Z,Rr | Косвенное сохранение | (Z) = Rr | – | 2 |
| ST | Z+,Rr | Косвенное сохранение с постинкрементом | (Z) = Rr, Z=Z+1 | – | 2 |
| ST | -Z,Rr | Косвенное сохранение с преддекрементом | Z=Z-1, (Z) = Rr | – | 2 |
| ST | Z+q,Rr | Косвенное сохранение относительное | (Z+q) = Rr | – | 2 |
| LPM | – | Загрузка из программной памяти в R0. Адрес – разряды 1-15 регистра Z, состояние разряда 0 определяет байт памяти: 0 – младший, 1 – старший. | R0 = (Z) | – | 3 |
| LPM | Rd,Z | Загрузка из программной памяти в Rd | Rd = (Z) | – | 3 |
| LPM | Rd,Z+ | Загрузка из программной памяти в Rd с постинкрементом | Rd = (Z), Z=Z+1 | – | 3 |
| SPM | нет | Сохранение в программной памяти | (Z) = R1:R0 | – | - |
| IN | Rd,P | Чтение регистра ввода/вывода | Rd = P | – | 1 |
| OUT | P,Rr | Запись в регистр ввода/вывода | P = Rr | – | 1 |
| PUSH | Rr | Сохранение в стеке | STACK = Rr | – | 2 |
| POP | Rd | Извлечение из стека | Rd = STACK | – | 2 |

Таблица 4. Команды сдвига и операции с битами

| **Мнемоника** | **Операнды** | **Описание** | **Действия** | **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 | Сдвиг влево через перенос | Rd(0)=C, Rd(n+1)=Rd(n), C=Rd(7) | Z,C,N,V,H,S | 1 |

| | | | | | |
|---|---|---|---|---|---|
| ROR | Rd | Сдвиг вправо через перенос | 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 | Обмен тетрадами | Rd(3..0) = Rd(7..4), Rd(7..4) = Rd(3..0) | – | 1 |
| BSET | s | Установка флага | SREG(s) = 1 | SREG(s) | 1 |
| BCLR | s | Сброс флага | SREG(s) = 0 | SREG(s) | 1 |
| SBI | P,b | Установка флага регистра ввода/вывода | I/O(P,b) = 1 | – | 2 |
| CBI | P,b | Сброс флага регистра ввода/вывода | I/O(P,b) = 0 | – | 2 |
| BST | Rr,b | Сохранение бита РОН в разряде SREG(T) | T = Rr(b) | T | 1 |
| BLD | Rd,b | Чтение бита из SREG(T) в РОН | 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 нулевого результата | Z = 1 | Z | 1 |
| CLZ | – | Сброс флага нулевого результата | Z = 0 | Z | 1 |
| SEI | – | Установка флага общего разрешения прерываний | I = 1 | I | 1 |
| CLI | – | Сброс флага общего разрешения прерываний | I = 0 | I | 1 |
| SES | – | Установка флага знака | S = 1 | S | 1 |
| CLN | – | Сброс флага знака | S = 0 | S | 1 |
| SEV | – | Установка флага переполнения | V = 1 | V | 1 |
| CLV | – | Сброс флага переполнения | V = 0 | V | 1 |
| SET | – | Установка флага копируемого бита | T = 1 | T | 1 |
| CLT | – | Сброс флага копируемого бита | T = 0 | T | 1 |
| SEH | – | Установка флага половинного переноса | H = 1 | H | 1 |
| CLH | – | Сброс флага половинного переноса | H = 0 | H | 1 |
| NOP | – | Нет операции (пустая команда) | нет | None | 1 |
| SLEEP | – | Режим энергосбережения | | None | 1 |
| WDR | – | Сброс сторожевого таймера | | | |

При описании команд использованы следующие обозначения:
Rd, Rr – регистры общего назначения с номерами d и r, Rr – источник, Rd - приёмник;

Rdh:Rdl – пара регистров, h – старший байт, l - младший;
K – константа (данные);
P,Ь – разряд b (Ь = O, …,7) порта P;
Rr(b) – разряд Ь (Ь = O, …,7) регистра Rr;
(XX (Y), (Z) – содержимое ячеек, адресуемых регистровыми парами X, Y, Z соответственно;
n – номер бита;
**S** – номер разряда в регистре SREG;
PC – содержимое программного счетчика;
к – приращение в счетчике команд (метка);
q – 6-разрядное смещение;
STACK – область памяти SRAM, адресуемая указателем стека SP;
C, Z, N, V, S, H, T, I – биты регистра состояния SREG;
d, r = 0…31 во всех случаях, кроме специально отмеченных.

## Приложение 2

## РЕГИСТРЫ ВВОДА/ВЫВОДА МИКРОКОНТРОЛЛЕРА AT90S8515

| Название | Функция | Адрес |
|---|---|---|
| ACSR | Регистр управления и состояния аналогового компаратора | $08 ($28) |
| ADCH | Регистр данных АЦП (старший байт) | — |
| ADCL | Регистр данных АЦП (младший байт) | — |
| ADCSR | Регистр управления и состояния АЦП | — |
| ADMUX | Регистр управления мультиплексором АЦП | — |
| ASSR | Регистр состояния асинхронного режима | — |
| DDRA | Регистр направления данных порта A | $1A ($3A) |
| DDRB | Регистр направления данных порта B | $17($37) |
| DDRC | Регистр направления данных порта C | $14 ($37) |
| DDRD | Регистр направления данных порта D | $11 ($31) |
| EEARH | Регистр адреса EEPROM (старший байт) | $1F ($3F) |
| EEARL | Регистр адреса EEPROM (младший байт) | $1E ($3E) |
| EECR | Регистр управления EEPROM | $1C ($3C) |
| EEDR | Регистр данных EEPROM | $1D ($3D) |
| GIFR | Общий регистр флагов прерываний | $3A ($5A) |
| GIMSK | Общий регистр маски прерываний | $3B($5B) |
| ICR1H | Регистр захвата таймера/счетчика 1 (старший байт) | $25 ($45) |
| ICR1L | Регистр захвата таймера/счетчика 1 (младший байт) | $24 ($44) |
| MCUCR | Общий регистр управления микроконтроллером | $35 ($55) |
| MCUSR | Регистр состояния микроконтроллера | — |
| OCR1AH | Регистр совпадения выхода A (старший байт) | $2B ($4B) |
| OCR1AL | Регистр совпадения выхода A (младший байт) | $2A ($4A) |
| OCR1BH | Регистр совпадения выхода B (старший байт) | $29 ($49) |
| OCR1BL | Регистр совпадения выхода B (младший байт) | $28 ($48) |
| OCR2 | Регистр совпадения выхода таймера/счетчика 2 | — |
| PINA | Выводы порта A | $19 ($39) |
| PINB | Выводы порта B | $16 ($36) |
| PINC | Выводы порта C | $13 ($36) |
| PIND | Выводы порта D | $10 ($30) |

| | | |
|---|---|---|
| PORTA | Регистр данных порта A | $1B($3B) |
| PORTB | Регистр данных порта B | $18 ($38) |
| PORTC | Регистр данных порта C | $15 ($38) |
| **Название** | **Функция** | **Адрес** |
| PORTD | Регистр данных порта D | $12($32) |
| SPCR | Регистр управления SPI | $0D ($2D) |
| SPDR | Регистр данных SPI | $0F ($2F) |
| SPH | Указатель стека (старший байт) | $3E ($5E) |
| SPL | Указатель стека (младший байт) | $3D ($5D) |
| SPSR | Регистр состояния SPI | $0E ($2E) |
| SREG | Регистр состояния | $3F ($5F) |
| TCCR0 | Регистр управления таймером/счетчиком 0 | $33 ($53) |
| TCCR1A | Регистр управления A таймером/счетчиком 1 | $2F ($4F) |
| TCCR1B | Регистр управления B таймером/счетчиком 1 | $2E ($4E) |
| TCCR2 | Счетный регистр таймера/счетчика 2 | — |
| TCNT0 | Счетный регистр таймера/счетчика 0 (8-разрядный) | $32($52) |
| TCNT1H | Счетный регистр таймера/счетчика 1 | $2D ($4D) |
| TCNT1L | Счетный регистр таймера/счетчика 1 (младший байт) | $2C ($4C) |
| TCNT2 | Счетный регистр таймера/счетчика 2 (8-разрядный) | — |
| TIFR | Регистр флагов прерываний от таймера/счетчика | $38 ($58) |
| TIMSK | Регистр маски прерываний от таймера/счетчика | $39 ($59) |
| UBRR | Регистр скорости передачи UART | $09 ($29) |
| UCR | Регистр управления UART | $0A ($2A) |
| UDR | Регистр данных UART | $0C ($2C) |
| USR | Регистр состояния UART | $0B ($2B) |
| WDTCR | Регистр управления сторожевым таймером | $21 ($41) |

# Приложение 3

## ДЕРЕКТИВЫ АССЕМБЛЕРА МИКРОКОНТРОЛЛЕРА AT90S8515

**.DEVICE – определяет тип целевого микроконтроллера.**

Директива DEVICE позволяет программисту указать, на каком микроконтроллере будет выполняться программа. Если в тексте программы указана эта директива, транслятор ассемблера будет проверять текст программы на наличие недопустимых операций (например, не поддерживаемых данным микроконтроллером). В случае попытки использования большего размера SRAM или EEPROM памяти, чем имеется у выбранного микроконтроллера, также будет выдано предупреждение. Если директива DEVICE отсутствует в тексте программы, то разрешены все команды семейства микроконтроллеров AVR, а размеры памяти не проверяются.

Симнтаксис:

.DEVICE AT90S1200 | AT90S2313 | AT90S2323 | AT90S2343 | AT90S4414 | AT90S8515 | ATMEGA103

**.CSEG – сегмент кода**

Директива CSEG определяет начало сегмента кода (программ). В исходном тексте программы может быть несколько сегментов кода. Транслятор ассемблера в процессе компиляции

программы объединяет все сегменты кода в один. Директива BYTE не может быть использована в сегменте кода. Если в программе нет явного указания названия сегмента, по умолчанию считается, что это сегмент кода. Директива CSEG не имеет никаких параметров. Сегмент кода имеет свой счетчик слов. Директива ORG может быть использована для размещения кода или констант в определенном програмистом месте памяти программ.

Синтаксис:

. CSEG

**.DSEG – сегмент данных.**

Директива DSEG определяет начало сегмента данных. В исходном тексте программы на ассемблере может быть несколько сегментов данных. В процессе трансляции все они будут объединены в один. Обычно сегмент данных содержит только директиву BYTE с метками. Сегмент данных имеет свой счетчик байтов. Директива ORG может быть использована для размещения переменных в конкретных местах SRAM. Директива DSEG не имеет параметров.

Синтаксис:

.DSEG

**.ESEG –EEPROM сегмент**

Директива ESEG определяет начало EEPROM сегмента. В исходном тексте программы может быть несколько EEPROM сегментов. В процессе трансляции все они будут объединены в один сегмент. Директива BYTE не может быть использована в EEPROM сегменте. Директива ESEG не имеет параметров. EEPROM сегмент имеет свой счетчик байтов. Директива ORG может быть использована для размещения кода или константы в определенном программном месте памяти EEPROM.

Синтаксис:

. ESEG

**.DB – определяет байты-констан.ты в памяти программ или EEPROM.**

Директива DB резервирует место в памяти программ или EEPROM. Для того чтобы иметь возможность обращаться к зарезервированному пространству, перед этой директивой следует ставить метку. Директива DB может быть расположена только в сегменте кода или EEPROM. Параметрами директивы DB является список выражений.

Список выражений представляет собой одно или несколько выражений, разделенных запятыми. Каждое выражение может быть равно числу от –128 до 255. Если выражение представляет собой отрицательное выражение, то оно будет помещено в память программ или EEPROM в дополнительном коде.

Синтаксис:

Метка: .DB список выражений

**.DEF- назначить регистру символьное имя.**

Директива DEF позволяет назначить регистру символьное имя, что позволяет сделать программу гораздо нагляднее и понятнее. Можно назначить одному регистру несколько символьных имен. Символьное имя регистра может быть переопределено в последующеим тексте программы.

Синтаксис:

.DEF символьное имя = регистр

**.DW- определение слов-констант в памяти программ или EEPROM.**

Директива DW резервирует место в памяти программ или EEPROM. Для того чтобы иметь возможность обращаться к зарезервированному пространству, перед этой директивой следует ставить метку. Директива DW может быть расположена только в сегменте кода или EEPROM. Параметрами директивы DW является список выражений.

Список выражений представляет собой одно или несколько выражений, разделенных запятыми. Каждое выражение может быть равно числу от –32768 до 65535. Если выражение представляет собой отрицательное выражение, то оно будет помещено в память программ или EEPROM в дополнительном коде.

Синтаксис:

Метка: .DW список выражений

**.BYTE – резервирует место (или несколько мест) размером 1 байт для переменной.**

Директива BYTE резервирует один байт в памяти SRAM для реализации переменной.

Для того, чтобы иметь возможность обращаться к этой переменной, перед директивой BYTE должна стоять метка. Директива имеет один параметр – количество байтов для резервирования. Директива может использоваться только для резервирования места в памяти данных (смотри директивы **CSEG, DSEG, ESEG**).

Синтаксис:

Метка: .BYTE числовое выражение.

**.EXIT – конец текста программы.**

Директива EXIT указывает транслятору ассемблера, что следует завершить трансляцию программы. При отсутствии этой директивы транслятор ассемблера работает до тех пор, пока исходный файл не закончится. Если директива EXIT встречается в файле, включаемом в текст директивой INCLUDE, транслятор ассемблера продолжает работу со строки, следующей после соответствующей директивы INCLUDE.

Синтаксис:

.EXIT

**.INCLUDE - вставить файл.**

Директива INCLUDE указывает транслятору ассемблера на необходимость вставить в исходный текст программы другой файл. Реально при обработке этой директивы транслируется файл, указанный в директиве INCLUDE, после завершения его обработки (при достижении конца файла или директивы EXIT) продолжается обработка основного файла. Вложенные файлы в свою очередь могут иметь директиву INCLUDE. Для облегчения понимания можно представить себе, что в текст программы вместо директивы INCLUDE вставляется соответствующий файл.

Синтаксис:

.INCLUDE “имя файла”

**.ORG – установить значения счетчика расположения.**

Директива ORG присваивает абсолютное значение счетчику. Параметром директивы является значение, которое должно быть присвоено счетчику. При использовании директивы ORG в сегменте данных будет определено значение, указывающее расположение в оперативной памяти SRAM. При использовании директивы ORG в сегменте кода будет определено значение, указывающее расположение в памяти программ. При использовании директивы ORG в EEPROM сегменте будет определено значение, указывающее расположение в памяти EEPROM.

Если перед директивой расположена метка (на этой же строке), то метка получит значение параметра директивы. Значение по умолчанию для сегмента кода и EEPROM равно 0, а для SRAM – 32 (т.к. регистры занимают пространство от 0 до 31). Обратите внимание, что для EEPROM и SRAM отсчитываются байты, в то время как в памяти программ – слова.

Синтаксис:

.ORG выражение

## Приложение 4

**ТАБЛИЦА ВЫБОРА КОНСТАНТ ДЛЯ ФОРМИРОВАНИЯ СКОРОСТИ ПЕРЕДАЧИ ДАННЫХ В КАНАЛЕ СВЯЗИ**

| Baud Rate | 1 MHz | %Error | 1.8432 MHz | %Error | 2 MHz | %Error | 2.4576 MHz | %Error |
|---|---|---|---|---|---|---|---|---|
| 2400 | UBRR= 25 | 0.2 | UBRR= 47 | 0.0 | UBRR= 51 | 0.2 | UBRR= 63 | 0.0 |
| 4800 | UBRR= 12 | 0.2 | UBRR= 23 | 0.0 | UBRR= 25 | 0.2 | UBRR= 31 | 0.0 |
| 9600 | UBRR= 6 | 7.5 | UBRR= 11 | 0.0 | UBRR= 12 | 0.2 | UBRR= 15 | 0.0 |
| 14400 | UBRR= 3 | 7.8 | UBRR= 7 | 0.0 | UBRR= 8 | 3.7 | UBRR= 10 | 3.1 |
| 19200 | UBRR= 2 | 7.8 | UBRR= 5 | 0.0 | UBRR= 6 | 7.5 | UBRR= 7 | 0.0 |
| 28800 | UBRR= 1 | 7.8 | UBRR= 3 | 0.0 | UBRR= 3 | 7.8 | UBRR= 4 | 6.3 |
| 38400 | UBRR= 1 | 22.9 | UBRR= 2 | 0.0 | UBRR= 2 | 7.8 | UBRR= 3 | 0.0 |
| 57600 | UBRR= 0 | 7.8 | UBRR= 1 | 0.0 | UBRR= 1 | 7.8 | UBRR= 2 | 12.5 |
| 76800 | UBRR= 0 | 22.9 | UBRR= 1 | 33.3 | UBRR= 1 | 22.9 | UBRR= 1 | 0.0 |
| 115200 | UBRR= 0 | 84.3 | UBRR= 0 | 0.0 | UBRR= 0 | 7.8 | UBRR= 0 | 25.0 |

| Baud Rate | 3.2768 MHz | %Error | 3.6864 MHz | %Error | 4 MHz | %Error | 4.608 MHz | %Error |
|---|---|---|---|---|---|---|---|---|
| 2400 | UBRR= 84 | 0.4 | UBRR= 95 | 0.0 | UBRR= 103 | 0.2 | UBRR= 119 | 0.0 |
| 4800 | UBRR= 42 | 0.8 | UBRR= 47 | 0.0 | UBRR= 51 | 0.2 | UBRR= 59 | 0.0 |
| 9600 | UBRR= 20 | 1.6 | UBRR= 23 | 0.0 | UBRR= 25 | 0.2 | UBRR= 29 | 0.0 |
| 14400 | UBRR= 13 | 1.6 | UBRR= 15 | 0.0 | UBRR= 16 | 2.1 | UBRR= 19 | 0.0 |
| 19200 | UBRR= 10 | 3.1 | UBRR= 11 | 0.0 | UBRR= 12 | 0.2 | UBRR= 14 | 0.0 |
| 28800 | UBRR= 6 | 1.6 | UBRR= 7 | 0.0 | UBRR= 8 | 3.7 | UBRR= 9 | 0.0 |
| 38400 | UBRR= 4 | 6.3 | UBRR= 5 | 0.0 | UBRR= 6 | 7.5 | UBRR= 7 | 6.7 |
| 57600 | UBRR= 3 | 12.5 | UBRR= 3 | 0.0 | UBRR= 3 | 7.8 | UBRR= 4 | 0.0 |
| 76800 | UBRR= 2 | 12.5 | UBRR= 2 | 0.0 | UBRR= 2 | 7.8 | UBRR= 3 | 6.7 |
| 115200 | UBRR= 1 | 12.5 | UBRR= 1 | 0.0 | UBRR= 1 | 7.8 | UBRR= 2 | 20.0 |

| Baud Rate | 7.3728 MHz | %Error | 8 MHz | %Error | 9.216 MHz | %Error | 11.059 MHz | %Error |
|---|---|---|---|---|---|---|---|---|
| 2400 | UBRR= 191 | 0.0 | UBRR= 207 | 0.2 | UBRR= 239 | 0.0 | UBRR= 287 | - |
| 4800 | UBRR= 95 | 0.0 | UBRR= 103 | 0.2 | UBRR= 119 | 0.0 | UBRR= 143 | 0.0 |
| 9600 | UBRR= 47 | 0.0 | UBRR= 51 | 0.2 | UBRR= 59 | 0.0 | UBRR= 71 | 0.0 |
| 14400 | UBRR= 31 | 0.0 | UBRR= 34 | 0.8 | UBRR= 39 | 0.0 | UBRR= 47 | 0.0 |
| 19200 | UBRR= 23 | 0.0 | UBRR= 25 | 0.2 | UBRR= 29 | 0.0 | UBRR= 35 | 0.0 |
| 28800 | UBRR= 15 | 0.0 | UBRR= 16 | 2.1 | UBRR= 19 | 0.0 | UBRR= 23 | 0.0 |
| 38400 | UBRR= 11 | 0.0 | UBRR= 12 | 0.2 | UBRR= 14 | 0.0 | UBRR= 17 | 0.0 |
| 57600 | UBRR= 7 | 0.0 | UBRR= 8 | 3.7 | UBRR= 9 | 0.0 | UBRR= 11 | 0.0 |
| 76800 | UBRR= 5 | 0.0 | UBRR= 6 | 7.5 | UBRR= 7 | 6.7 | UBRR= 8 | 0.0 |
| 115200 | UBRR= 3 | 0.0 | UBRR= 3 | 7.8 | UBRR= 4 | 0.0 | UBRR= 5 | 0.0 |

$$BAUD = \frac{f_{CK}}{16(UBRR + 1)}$$

- BAUD = Baud rate
- $f_{CK}$ = Crystal Clock frequency
- UBRR = Contents of the UART Baud Rate register, UBRR (0 - 255)

## Приложение 5

**АЛЬТЕРНАТИВНЫЕ ФУНКЦИИ PORTB И PORTD МИКРОКОНТРОЛЛЕРААТ90S8515**

| Пины порта B | Альтернативные функции | Пины порта D | Альтернативные функции |
|---|---|---|---|
| PB0 (T0) | внешний вход таймер/счетчика 0 | PD0 (RXD) | вход данных UART |

| | | | |
|---|---|---|---|
| PB1 (T1) | внешний вход таймер/счетчика 1 | PD1 (TXD) | выход данных UART |
| PB2 (AIN0) | положительный вход аналогового компаратора | PD2 (INT0) | вход внешнего прерывания 0 |
| PB3 (AIN)1 | отрицательный вход аналогового компаратора | PD3 (INT1) | вход внешнего прерывания 1 |
| PB4 (SS) | | PD4 | |
| PB5 (MOSI) | вход данных для SPI | PD5 (OC1A) | выход совпадения с А таймер/счетчика 1 |
| PB6 (MISO) | выход данных для SPI | PD6 (WR) | вход импульса записи |
| PB7 (SCK) | Вход тактовых импульсов SPI | PD7 (RD) | 0 – программирование, 1 – чтение команд |

## Приложение 6

### ПЕРЕЧЕНЬ КОМАНД АССЕМБЛЕРА МИКРОКОНТРОЛЛЕРА СЕМЕЙСТВА MCS51

***Список условных обозначений***

R*n* – регистры общего назначения (*n* = 0, 1,…,7);
R*i* – указатель косвенного адреса (регистры R0 или R1);
ad – адрес ячейки внутренней памяти данных (00H, 01H,…, 7FH);
adr16 – шестнадцатибитовый адрес (имя метки);
d – восьмибитовая константа;
d16 – шестнадцатибитовая константа;
bit – бит в резидентной памяти данных или в регистрах специальных функций;
rel – восьмибитовый адрес при переходе в пределах страницы (имя метки);
Тк – количество машинных циклов выполнения команды.

***Список команд, влияющих на флаги в регистре PSW***

| Команда | C | OV | AC | Команда | C | OV | AC |
|---|---|---|---|---|---|---|---|
| ADD | x | x | x | CLR C | 0 | – | – |
| ADDC | x | x | x | CPL C | x | – | – |
| SUBB | x | x | x | ANL C, bit | x | – | – |
| MUL | 0 | x | – | ANL C, /bit | x | – | – |
| DIV | 0 | x | – | ORL C, bit | x | – | – |
| DA | x | – | – | ORL C, /bit | x | – | – |
| RRC | x | – | – | MOV C, bit | x | – | – |
| RLC | x | – | – | CJNE | x | – | – |
| SETB C | 1 | – | – | – | – | – | – |

C – флаг переноса, установка и сброс аппаратно и программно;
AC – флаг дополнительного переноса, установка и сброс аппаратно при выполнении команд сложения и вычитания;
OV – флаг переполнения, установка и сброс аппаратно;

«x» – содержимое флага устанавливается по результату выполнения команды;
«–» – команда не влияет на значение флага.

***Команды передачи данных***

| Мнемоника | | Тк | Мнемоника | | Тк | Мнемоника | | Тк |
|---|---|---|---|---|---|---|---|---|
| MOV | A,R*n* | 1 | MOV | ad,@R*i* | 2 | MOVX | @R*i*,A | 2 |
| MOV | A,ad | 1 | MOV | ad,#d | 2 | MOVX | @DPTR,A | 2 |
| MOV | A,@R*i* | 1 | MOV | @R*i*,A | 1 | PUSH | ad | 2 |
| MOV | A,#d | 1 | MOV | @R*i*,ad | 2 | POP | ad | 2 |
| MOV | R*n*,A | 1 | MOV | @R*i*,#d | 1 | XCH | A,R*n* | 1 |
| MOV | R*n*,ad | 2 | MOV | DPTR,#d16 | 2 | XCH | A,ad | 1 |
| MOV | R*n*,#d | 1 | MOVC | A,@A+DPTR | 2 | XCH | A,@R*i* | 1 |
| MOV | ad,A | 1 | MOVC | A,@A+PC | 2 | XCHD | A,@R*i* | 1 |
| MOV | ad,R*n* | 2 | MOVX | A,@R*i* | 2 | SWAP | A | 1 |
| MOV | $ad_1$,$ad_2$ | 2 | MOVX | A,@DPTR | 2 | | | |
| *Команды арифметических операций* | | | | | | | | |
| ADD | A,R*n* | 1 | SUBB | A,R*n* | 1 | INC | A | 1 |
| ADD | A,ad | 1 | SUBB | A,ad | 1 | INC | R*n* | 1 |
| ADD | A,@R*i* | 1 | SUBB | A,@R*i* | 1 | INC | ad | 1 |
| ADD | A,#d | 1 | SUBB | A,#d | 1 | INC | @R*i* | 1 |
| ADDC | A,R*n* | 1 | DEC | A | 1 | INC | DPTR | 2 |
| ADDC | A,ad | 1 | DEC | R*n* | 1 | DA | A | 1 |
| ADDC | A,@R*i* | 1 | DEC | Ad | 1 | MUL | AB | 4 |
| ADDC | A,#d | 1 | DEC | @R*i* | 1 | DIV | AB | 4 |
| *Команды логических операций* | | | | | | | | |
| ANL | A,R*n* | 1 | ORL | A,R*n* | 1 | XRL | A,R*n* | 1 |
| ANL | A,ad | 1 | ORL | A,ad | 1 | XRL | A,ad | 1 |
| ANL | A,@R*i* | 1 | ORL | A,@R*i* | 1 | XRL | A,@R*i* | 1 |
| ANL | A,#d | 1 | ORL | A,#d | 1 | XRL | A,#d | 1 |
| ANL | ad,A | 1 | ORL | ad,A | 1 | XRL | ad,A | 1 |
| ANL | ad,#d | 2 | ORL | ad,#d | 2 | XRL | ad,#d | 2 |
| CLR | A | 1 | RL | A | 1 | RR | A | 1 |
| CPL | A | 1 | RLC | A | 1 | RRC | A | 1 |
| *Команды операций с битами* | | | | | | | | |
| CLR | C | 1 | CPL | C | 1 | ORL | C,bit | 2 |
| CLR | bit | 1 | CPL | Bit | 1 | ORL | C, /bit | 2 |
| SETB | C | 1 | ANL | C,bit | 2 | MOV | C,bit | 1 |
| SETB | bit | 1 | ANL | C, /bit | 2 | MOV | bit,C | 2 |
| *Команды передачи управления* | | | | | | | | |
| NOP | | 1 | JZ | Rel | 2 | DJNZ | R*n*,rel | 2 |
| CALL | adr16 | 2 | JNZ | Rel | 2 | DJNZ | ad,rel | 2 |
| RET | | 2 | JC | Rel | 2 | CJNE | A,ad,rel | 2 |
| RETI | | 2 | JNC | Rel | 2 | CJNE | A,#d,rel | 2 |
| JMP | adr16 | 2 | JB | bit,rel | 2 | CJNE | R*n*,#d,rel | 2 |
| SJMP | rel | 2 | JNB | bit,rel | 2 | CJNE | @R*i*,#d,rel | 2 |
| JMP | @A+DPTR | 2 | JBC | bit,rel | 2 | | | |

## Приложение 7

**Образец титульной страницы отчёта**

**Российский государственный гидрометеорологический университет**

**КАФЕДРА МОРСКИХ ИНФОРМАЦИОННЫХ СИСТЕМ**

**Лабораторная работа №**

«Название работы»

Выполнил (а) студент (ка) гр. №_____ Фамилия И.О.

Проверил (а) преподаватель Фамилия И.О.

Санкт-Петербург

20___г.

## Литература


**а) Основная литература:**

1. *Новожилов О.П.* Основы микропроцессорной техники. Учебное пособие в двух томах. Т.1. – М.: ИП РадиоСофт, 2007. – 432 с.
2. *Микушин А.*В., Сажнев А.М, Сединин В.И. Цифровые устройства и микропроцессоры. Учебное пособие. – СПб.: БХВ-Петербург, 2010. – 832 с.
3. Однокристальная микро-ЭВМ семейства MCS51 Методические указания к лабораторным работам по дисциплине «Цифровые и микропроцессорные устройства» / Сост.: *А.К. Артемьев, А.В. Матвеев, И.С. Минченко, Ю.В.Сентябрёв.* СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2000. – 32 с.

**б) дополнительная литература:**

4. *Бойко В.И.* и др. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры. − СПб.: БХВ-Петербург, 2004. – 455 с.

5. *Водовозов А.М.* Микроконтроллеры для систем автоматики. Учебное пособие. − Вологда. ВоГТУ, 2002. – 123 с.
6. *Артемьев А.К., Матвеев А.В., Минченко И.С., Сентябрев Ю.В.* Микропроцессоры в автоматизированных системах контроля и управления РЭС Учеб. пособие. СПб.: Изд-во СПбГЭТУ "ЛЭТИ", 2003. – 60 с.
7. Гладштейн М.А. Микроконтроллеры смешанного сигнала C8051Fxxx фирмы Silicon Laboratories и их применение. − М.: Издательский дом «Додэка-XXI», 2008. – 328 с.

**Оглавление**

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

«Микроконтроллеры»

Владимир Алексеевич Большаков
Виктория Александровна Миклуш

*Большаков В.А., Миклуш В.А* „.Микроконтроллеры. СПб: ООО "Андреевский издательский дом", СПб.: 2017.- 80 с.