Лабораторная работа 1

ОСНОВЫ РАБОТЫ В СРЕДЕ ATMEL STUDIO. АРХИТЕКТУРА И СИСТЕМА КОМАНД МИКРОКОНТРОЛЛЕРА ATMEGA32

Цель работы

Изучение основ программирования микроконтроллеров семейства Atmel ATmega в среде разработки Atmel Studio. Знакомство с системой команд микроконтроллера ATmega32.

Теоретические сведения

Возможности отладочной платы EasyAVR v7

Отладочные платы EasyAVR предоставляют возможность проектировать программно-аппаратные средства, основывающиеся на микроконтроллерах Atmel AVR. Семейство отладочных плат EasyAVR совершенствовалось в течение нескольких поколений и на сегодняшний день имеет множество пользователей среди студентов, любителей и профессионалов в области разработки микропроцессорных систем.

В состав отладочной платы EasyAVR v7 включены следующие компоненты и интерфейсы:

– программатор mikroProg;

– кварцевый резонатор;

– четыре порта ввода-вывода, при этом каждый вывод снабжён кнопкой, светодиодом и выводом к нескольким внешним разъёмам;

– интерфейс UART, выведенный к портам RS-232 и USB;

– микросхема EEPROM, подключённая к микроконтроллеру через интерфейс I_2C;

– четыре семисегментных индикатора;

– имитатор аналогово входного сигнала;

– динамик Piezo Buzzer;

– интерфейс для подключения 2x16 символьного ЖК-дисплея;

– интерфейс для подключения графического дисплея 128x64;

– интерфейс для подключения сенсорной панели;

– интерфейсы для подключения температурных датчиков;

– интерфейсы для подключения плат-расширений (GPS, GSM, карта памяти microSD, инфракрасный порт, интерфейс Wi-Fi и т.д.).

Отладочная плата EasyAVR v7 снабжена программатором mikroProg, позволяющим «перепрашивать» 65 различных AVR микроконтроллеров фирмы Atmel. Программатор подключается к персональному компьютеру через порт USB 2.0. Текущее состояние программатора показывается тремя светодиодными индикаторами. Светодиод «Link» загорается, когда установлена USB-связь с персонального компьютера. Светодиод «Active» светится, когда выполняется программа. Светодиод «Data» загорается при передаче данных между программатором и программным обеспечением AVRFlash на персональном компьютере.

Одним из главных достоинств EasyAVR v7 являются четыре группы 8-ми разрядных портов ввода/вывода. Все выводы портов оснащены кнопками и светодиодными индикаторами. Для каждого порта на плате размещены три 8-ми разрядных разъёма для подключения внешних устройств.

Для возможности взаимодействия с различными современными компьютерами плата оснащена разъёмом RS-232 и дополнительным разъёмом USB, подключёнными к интерфейсу UART. Микросхема FT232RL позволяет конвертировать UART сигналы в соответствии со стандартами USB. Для использования связи USB-UART требуется установка драйвера FTDI на компьютер.

Плата оснащена интерфейсом I_2C, который используется для подключения периферийных низкоскоростных устройств. Всего к одной шине могут быть подключены до 112 устройств. Каждое устройство должно иметь уникальный адрес. В состав платы входит микросхема EEPROM, подключённая к интерфейсу I_2C, имеющая 1024 байта доступной памяти при скорости передачи данных 400 кГц.

Плата оснащена четырьмя семисегментными индикаторами, которые могут быть использованы для отображения цифр от 0 до 9, разделителя в виде точки и некоторых букв.

Плата EasyAVR v7 содержит интерфейс в виде двух потенциометров для имитации аналоговых входных напряжений, которые могут быть использованы на любом из двенадцати поддерживающих аналоговые входные сигналы контактов.

Микроконтроллеры могут создавать звук при помощи широтно-импульсной модуляцией (PWM), подавая формируемый сигнал на

встроенный Piezo Buzzer. Резонансная частота Piezo Buzzer равна 3.8 кГц. Возможно создание звука в диапазоне от 2 до 4 кГц.

Микроконтроллер Atmel AVR ATmega32

По своим функциональным возможностям современные микроконтроллеры представляют собой полномасштабный компьютер (процессор, память программ и память данных, порты ввода-вывода, таймеры и периферийные устройства), выполненный на одном кристалле. Все необходимые для функционирования компоненты находятся внутри микросхемы, поэтому для работы микроконтроллера требуется только подача внешнего питания. В зависимости от рабочей частоты, модели и используемых периферийных устройств микроконтроллер потребляет от 5 до 200 мА.

Микроконтроллер реализовывает гарвардскую архитектуру, т.е. имеет энергонезависимую память программ, и память данных, которая сбрасывается при отключении питания. Энергонезависимая память может быть OTP (one-time programmable, записывается при производстве микроконтроллера), но в большинстве случаев используется EEPROM, что позволяет изменять исполняемую программу. Программа записывается в микроконтроллер с помощью специального устройства – программатора. Записываемый машинный код называется firmware, или «прошивкой».

Помимо процессора и памяти в микроконтроллере могут присутствовать следующие периферийные устройства: модуль USART, аналого-цифровой преобразователь, цифро-аналоговый преобразователь, таймер, аналоговый компаратор, интерфейсы SPI, CAN и другие. Микроконтроллеры отличаются по своим характеристикам: форм-фактору, количеству выводов, объёму памяти программ и данных, видам и количеству периферийных устройств, тактовой частоте и другим параметрам.

Подробную документацию по схемотехнике и организации каждого конкретного микроконтроллера можно получить из спецификации (datasheet) производителя.

Микроконтроллер ATmega32 является маломощным 8-разрядным КМОП микроконтроллером на основе усовершенствованной архитектуры RISC, выполняющий большинство инструкций за один такт.

Ядро ATmega32 сочетает в себе богатый набор команд и 32 регистра общего назначения. Все 32 регистра подключены непосредственно к

арифметико-логическому устройству (АЛУ), что позволяет инструкциям за один такт обращаться к двум независимым регистрам. Такая архитектура позволяет достигать пропускной способности до десяти раз быстрее по сравнению с обычными CISC микроконтроллерами. Блок-схема микроконтроллера приведена на рисунке 1.

ATmega32 предоставляет следующие возможности:

– 131 инструкция, большинство выполняемых за один такт

– пропускная способность до 16 MIPS при частоте до 16 МГц

– 32 КБ In-System Pragrammable (ISP) памяти программ с возможностью Read-While-Write;

– 1024 байт EEPROM;

– 2 КБ внутренней SRAM;

– 32 программируемых линии портов ввода/вывода общего назначения;

– 32 восьми разрядных регистра общего назначения;

– внешние и внутренние источники прерываний;

– интерфейс JTAG;

– два 8-разрядных таймеров-счётчиков с отдельными делителями частоты и режимами сравнения;

– один 16-разрядный таймер-счётчик с отдельным предварительным делителем, режимами сравнения и захвата;

– счётчик реального времени с отдельным генератором;

– программируемый Watchdog-таймер с встроенный генератором;

– четыре канала широтно-импульсной модуляции (ШИМ);

– программируемый последовательный интерфейс USART;

– последовательный интерфейс SPI (Master/Slave);

– встроенный аналоговый компаратор;

– 8-канальный 10-битовый АЦП с программируемым коэффициентом усиления;

– байт-ориентированный двухпроводной последовательный интерфейс;

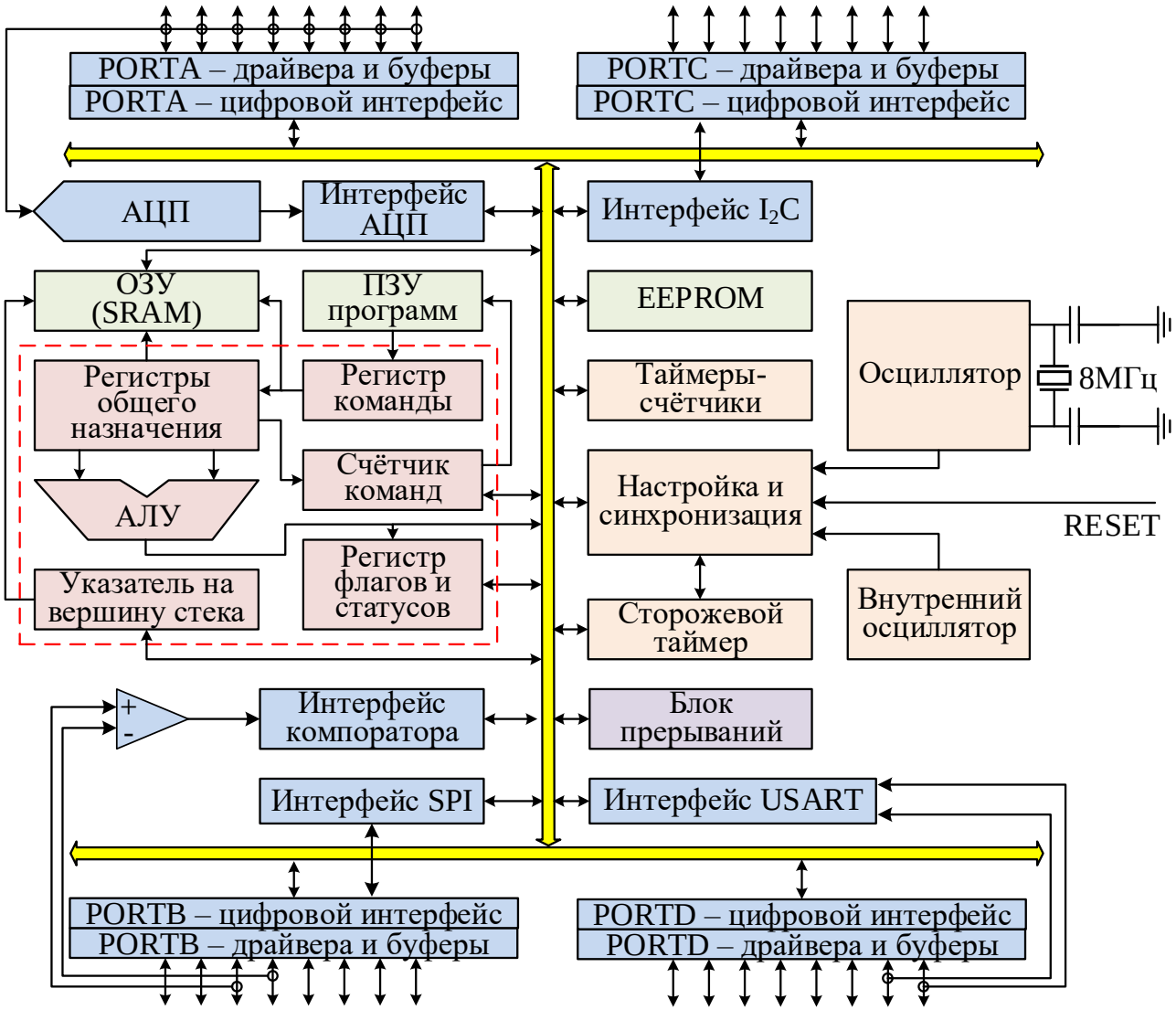
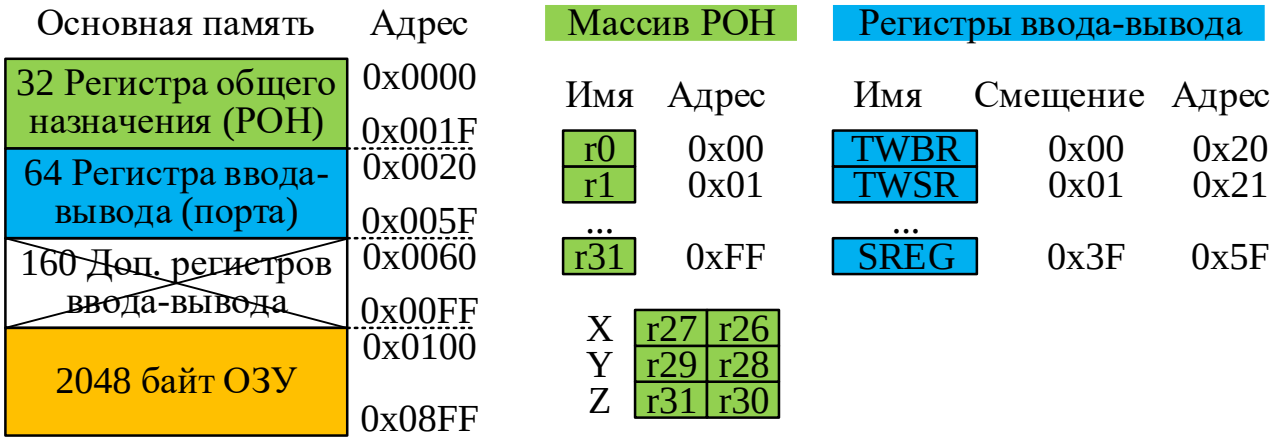


Рис. 1. блок-схема микроконтроллера Atmel AVR ATmega32

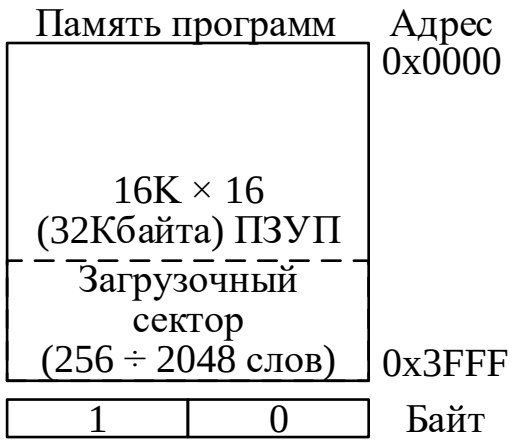
Встроенная ISP память позволяет перепрограммировать микроконтроллер через последовательный SPI интерфейс с помощью обычного программатора энергонезависимой памяти или с помощью встроенной программы загрузчика из ядра AVR. Программа загрузки может использовать любой интерфейс для загрузки приложения в память программ.

Адресное пространство МК

В МК используются два отдельных адресных пространства – одно для оперативной памяти (регистры общего назначения, регистры ввода-вывода, ОЗУ), второе для постоянной памяти программ (ПЗУП).



Оперативная память (данные)



Постоянная память программ

Регистр флагов и состояний SREG

Номер разряда	7	6	5	4	3	2	1	0
Имя флага	I	T	H	S	V	N	Z	C

Назначение разрядов регистра SREG:

Номер разряда	Обозначение	Имя флага	Описание
0	C	Carry flag	Флаг переноса. Устанавливается, если во время выполнения операции был перенос из старшего разряда результата
1	Z	Zero flag	Флаг нулевого результата. Устанавливается, если результат операции равен 0
2	N	Negative flag	Флаг отрицательного результата. Устанавливается, если MSB (Most Significant Bit - старший бит) результата равен 1 (правильно показывает знак результата, если не было переполнения разрядной сетки знакового числа)
3	V	Two's complement overflow	Флаг переполнения дополнения до двух. Устанавливается, если во время выполнения операции было переполнение разрядной сетки знакового результата

4	S	Sign flag	Бит знака, S = N XOR V. Бит S всегда равен исключающему ИЛИ между флагами N (отрицательный результат) и V (переполнение дополнения до двух). Правильно показывает знак результата и при переполнении разрядной сетки знакового числа
5	H	Half Carry flag	Флаг половинного переноса. устанавливается, если во время выполнения операции был перенос из 3-го разряда результата
6	T	Transfer bit	Хранение копируемого бита. Команды копирования битов BLD (Bit LoaD) и BST (Bit STore) используют этот бит как источник и приёмник обрабатываемого бита. Бит из регистра регистрового файла может быть скопирован в T командой BST, бит T может быть скопирован в бит регистрового файла командой BLD
7	I	Global interrupt flag	Общее разрешение прерываний. Для разрешения прерываний этот бит должен быть установлен в единицу. Управление отдельными прерываниями производится регистром маски прерываний - GIMSK/TIMSK. Если флаг сброшен (0), независимо от состояния GIMSK/TIMSK, прерывания не разрешены. Бит I очищается аппаратно после входа в прерывание и восстанавливается командой RETI, для разрешения обработки следующих прерываний

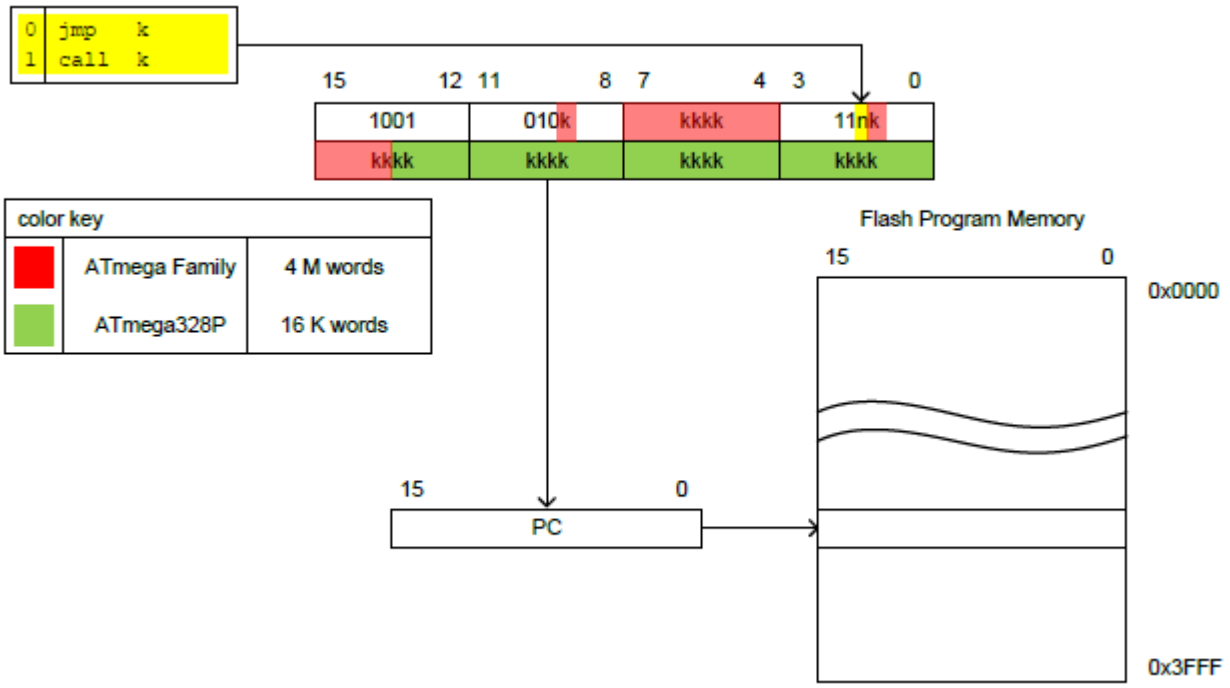
Форматы команд МК AVR ATmega32

Условное обозначение:
PC – счётчик команд
r и d – разряд адреса (номера) регистров общего назначение
A – разряд номера (адреса) регистров ввода-вывода (портов)
k – разряд адреса (при прямой адресации)
K – разряд константы (при непосредственной адресации)
q – разряд константы (при адресации со смещением)
b – разряд в регистре общего назначения или регистре ввода-вывода
s – разряд в регистре флагов и состояния SREG

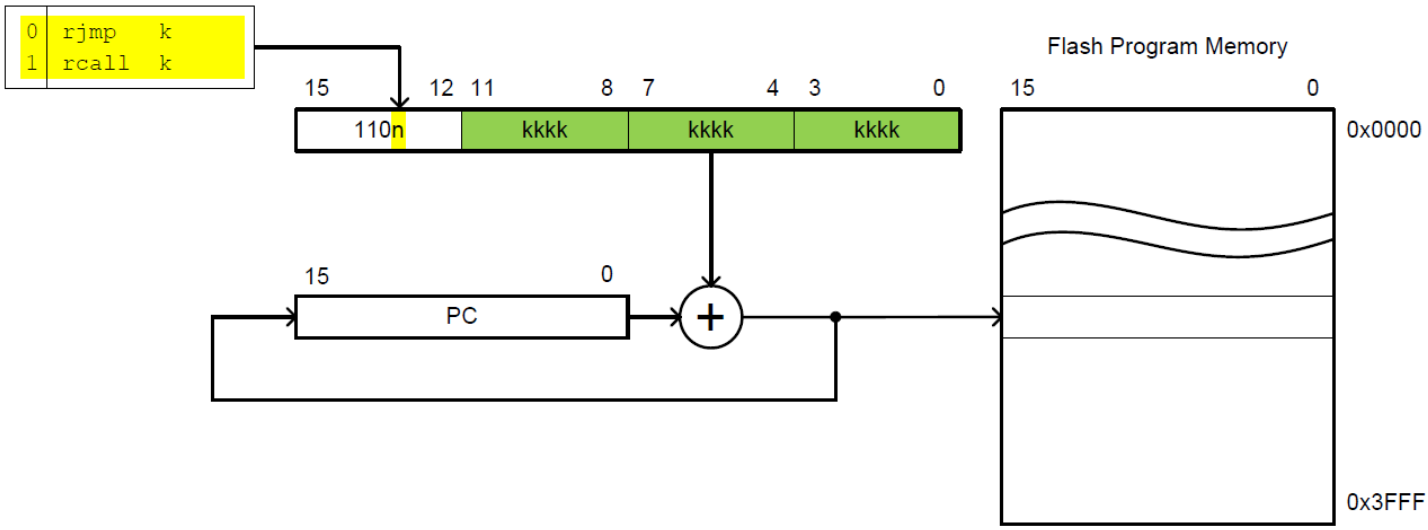
Диапазоны допустимых значений (по умолчанию):
$r \in [0;31]$
$d \in [0;31]$
$A \in [0;63]$
$s \in [0;7]$
$b \in [0;7]$

Команды передачи управления

Безусловные с прямой адресацией

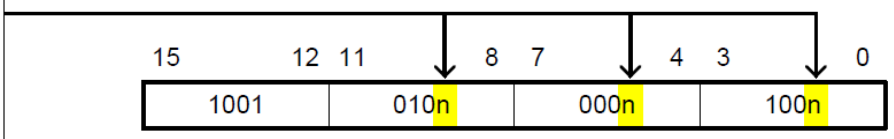


Безусловные с относительной адресацией

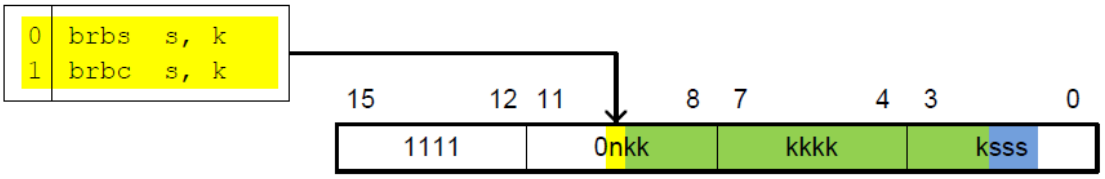


Безусловные с косвенно-регистровой адресацией

nnn	opcode	notes
000		
001	ijmp	see illustration
010		
011	eijmp	n/a to 328P
100	ret	PC ← M[SP + 1]
101	icall	see illustration
110	reti	PC ← M[SP + 1]
111	eicall	n/a to 328P



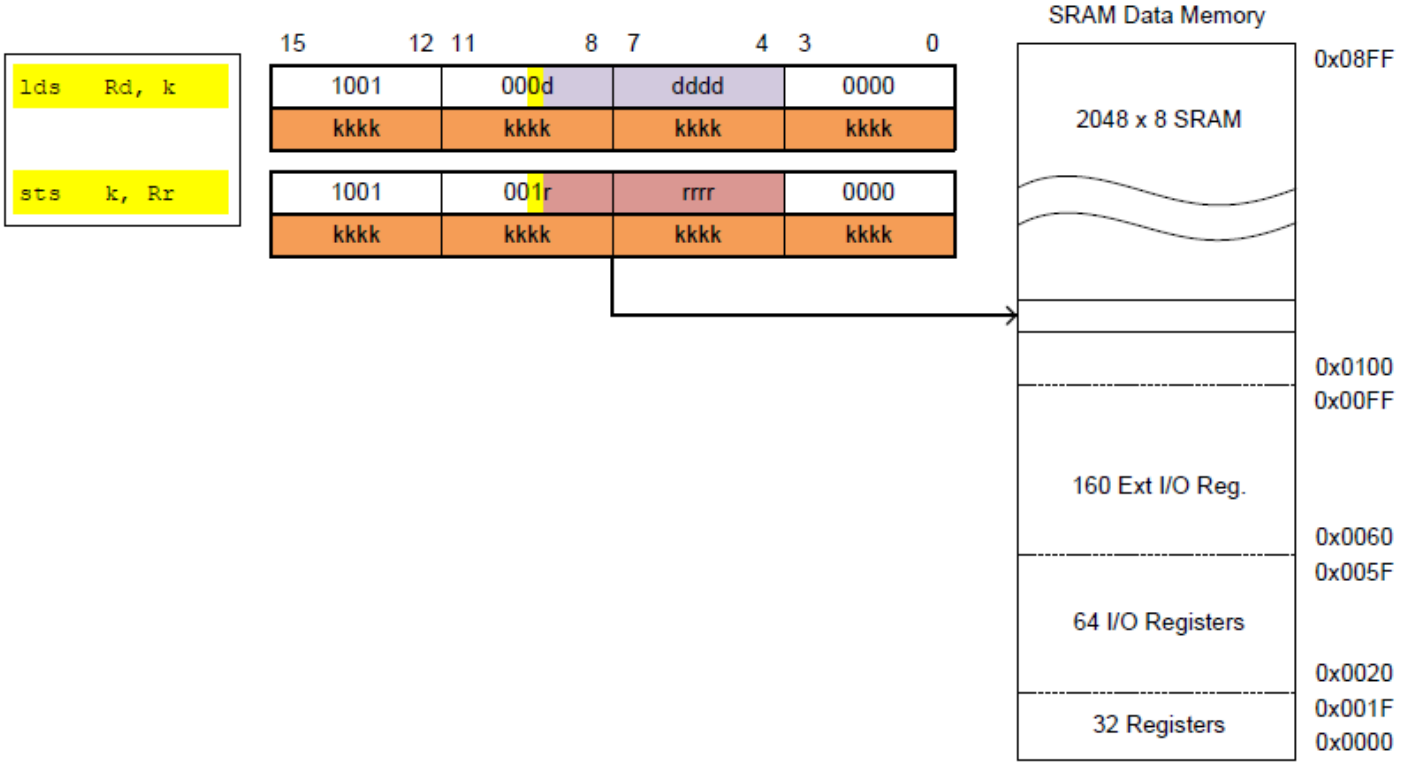
Условные (с относительной адресацией)



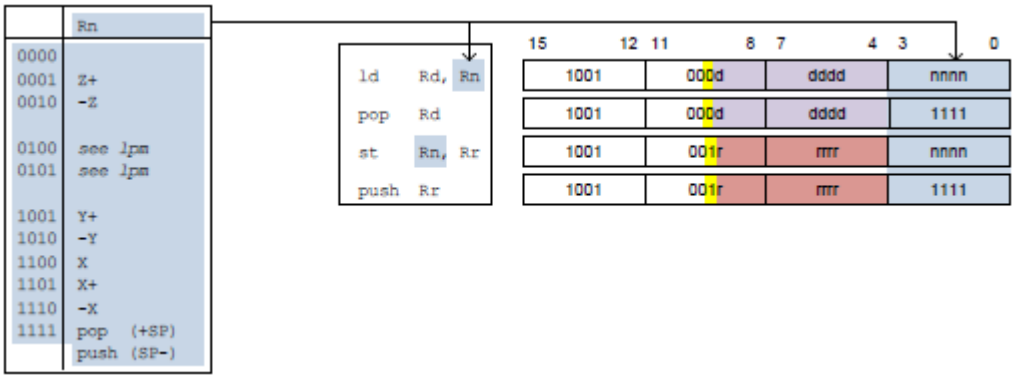
alias			
SREG	sss	brbs s, k	brbc s, k
I	111	brie k	brid k
T	110	brts k	brtc k
H	101	brhs k	brhc k
S	100	brlt k	brge k
V	011	brvs k	brvc k
N	010	brmi k	brpl k
Z	001	breq k	brne k
C	000	brcs k brlo k	brcc k brsh k

Команды передачи данных

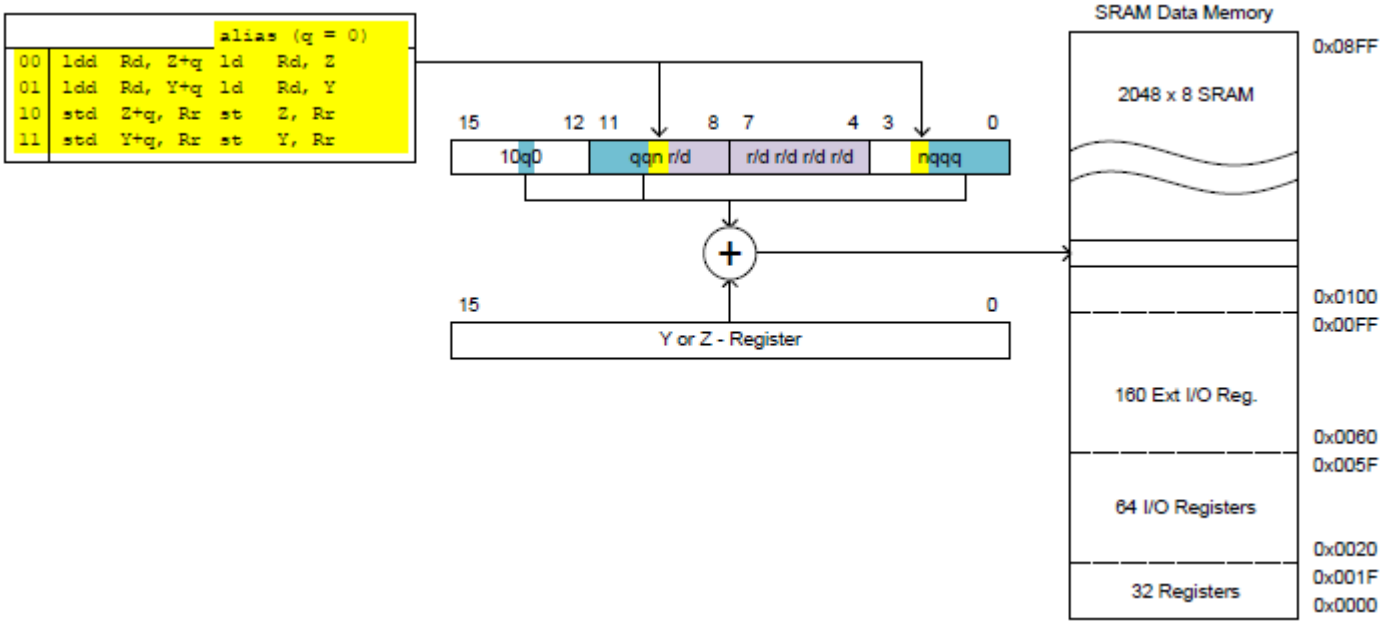
С прямой адресацией



С косвенно-регистровой адресацией

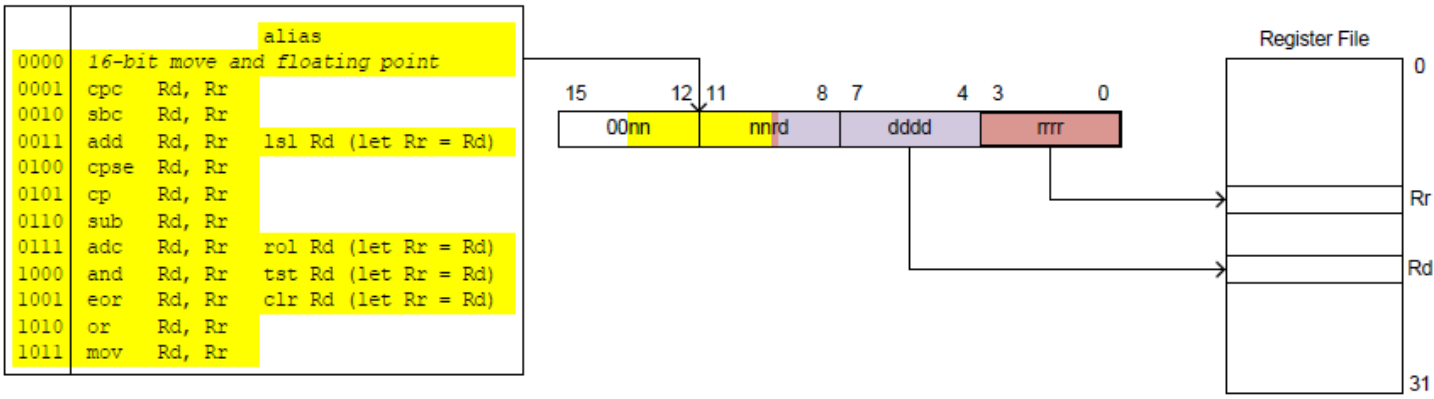


С косвенно-регистровой адресацией и смещением

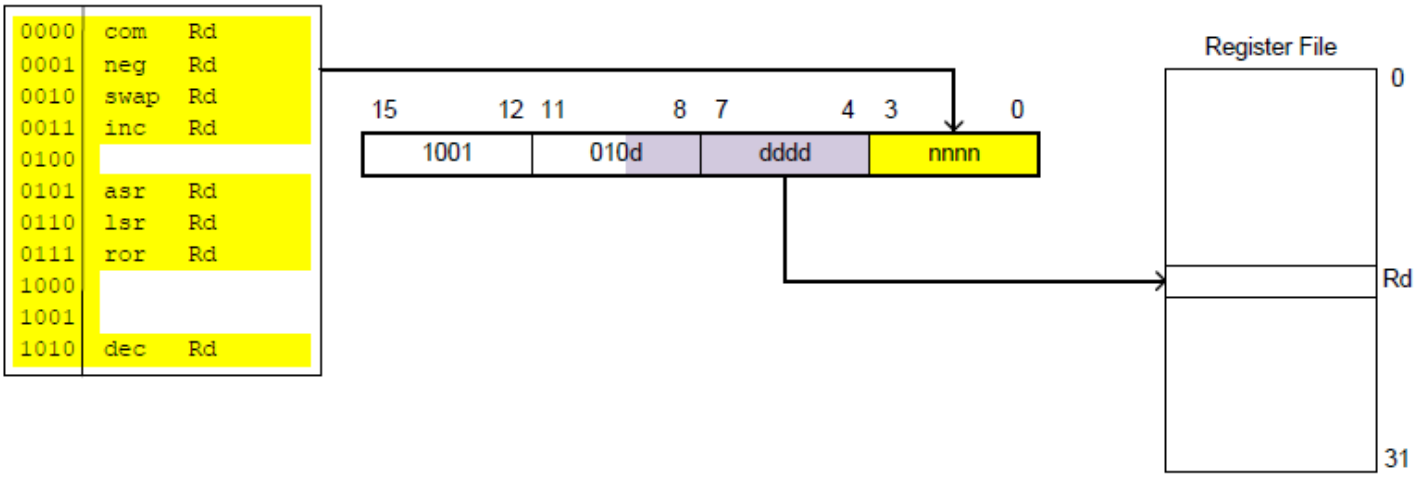


Арифметические и логические команды

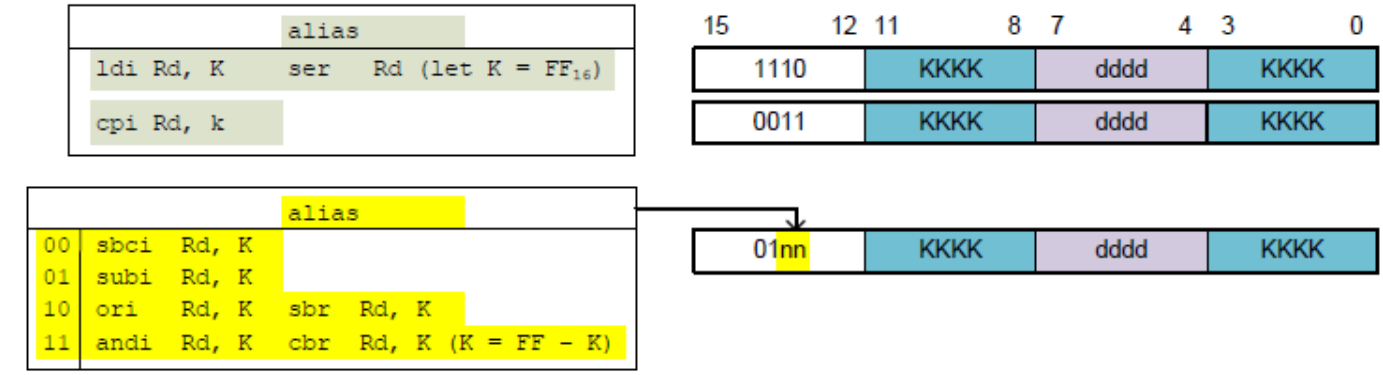
Бинарные арифметические и логические команды



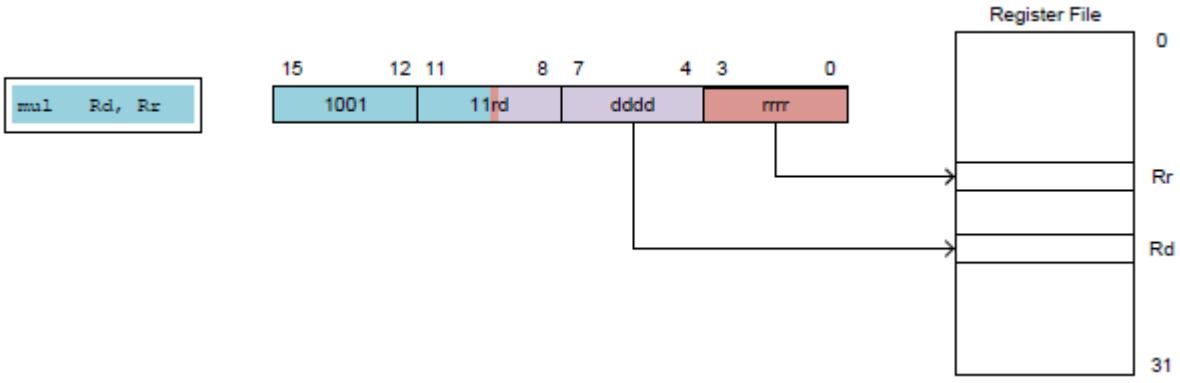
Унарные команды (безоперандные)



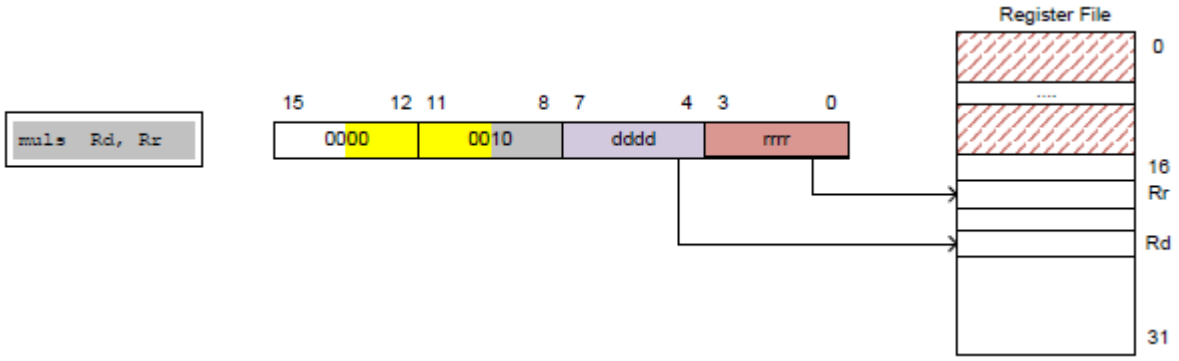
Унарные команды (с операндами)



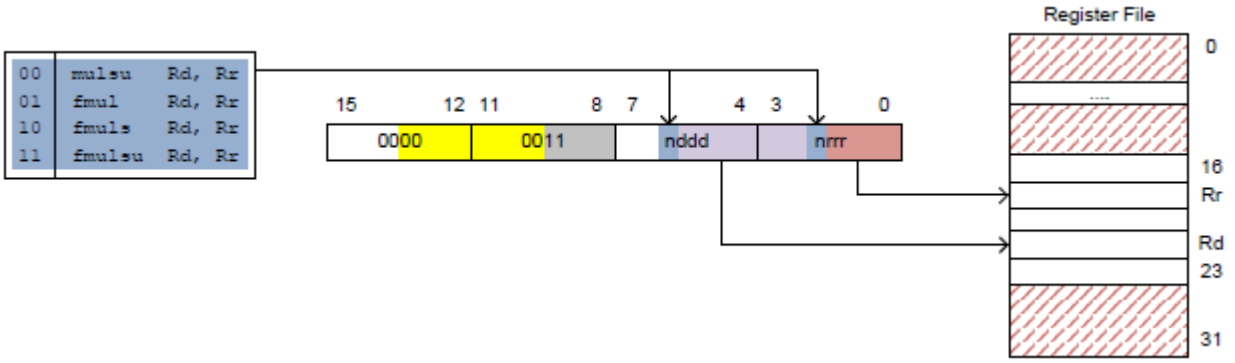
Беззнаковое умножение



Умножение чисел со знаком

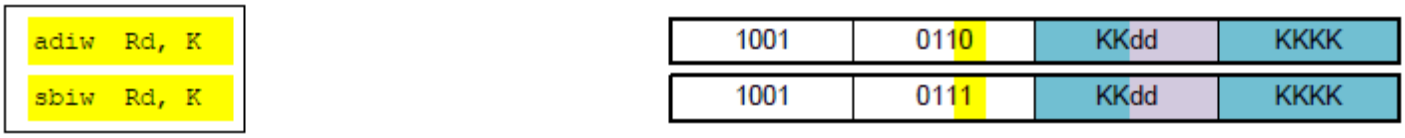


Умножение смешанных (беззнаковое с числом со знаком) чисел и дробное умножение

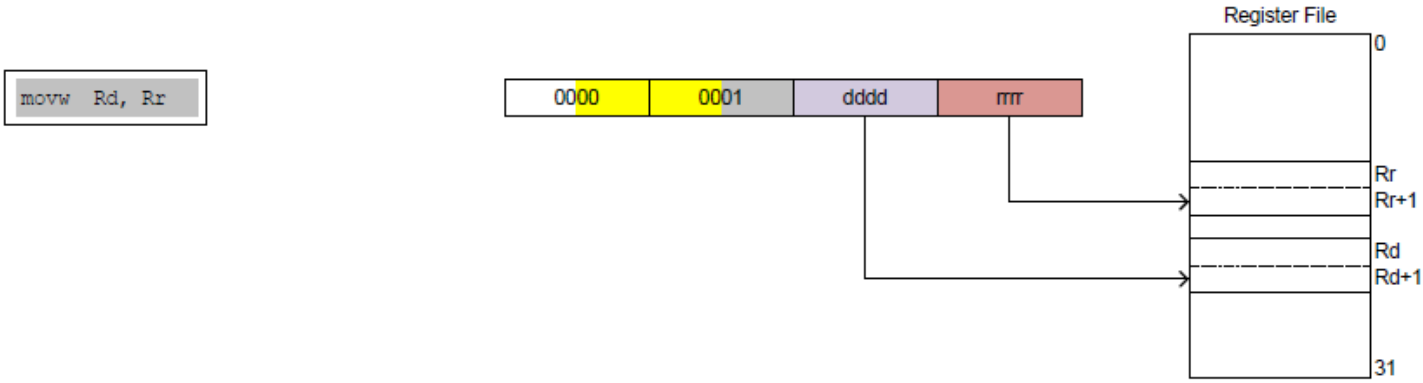


Команда для обработки 16-разрядных слов

Арифметические

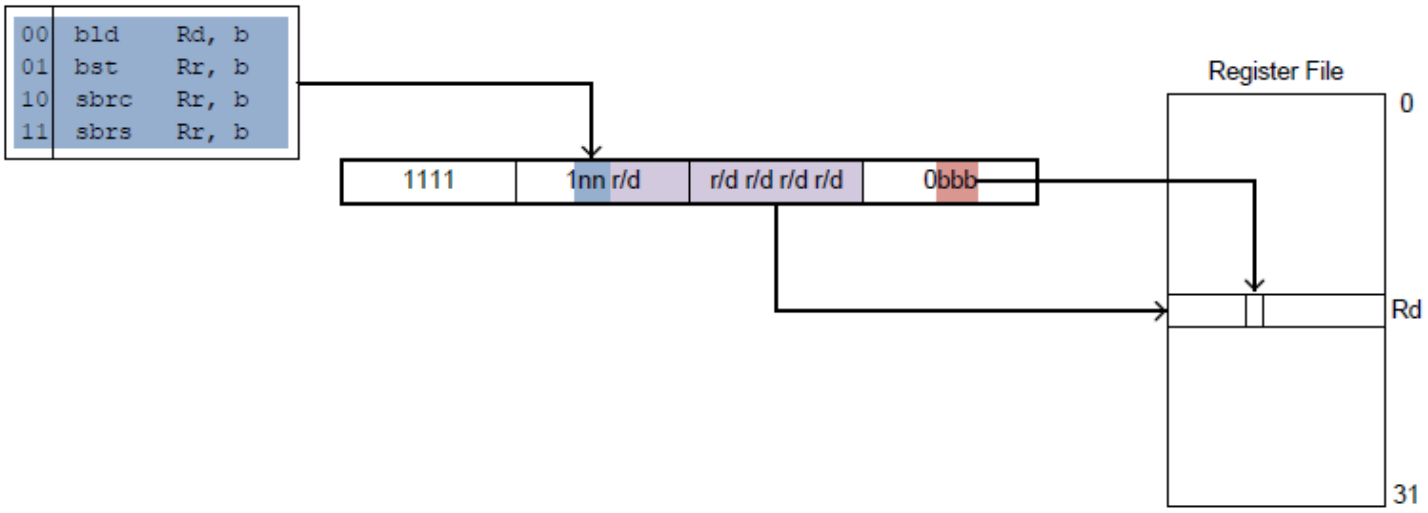


Пересылка

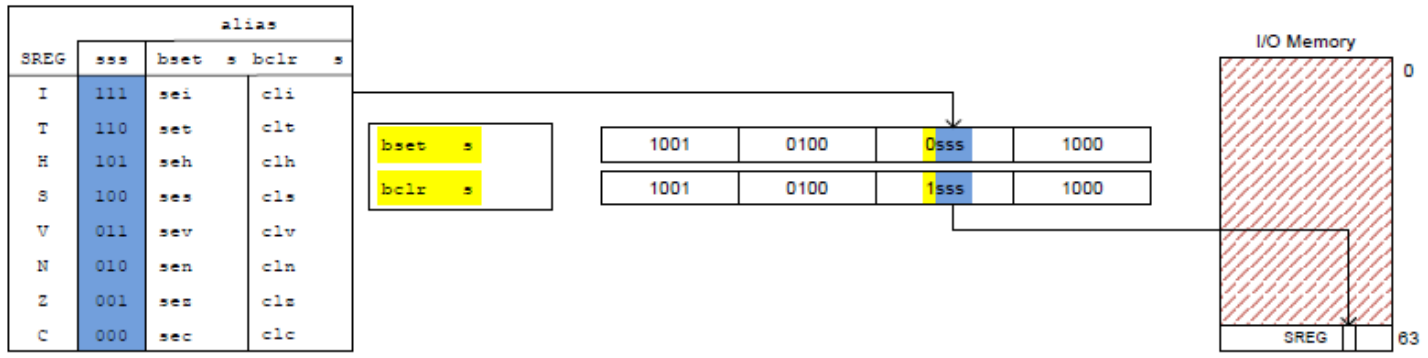


Битовые команды

Команды установки/очистки/управления по отдельным разрядам регистров

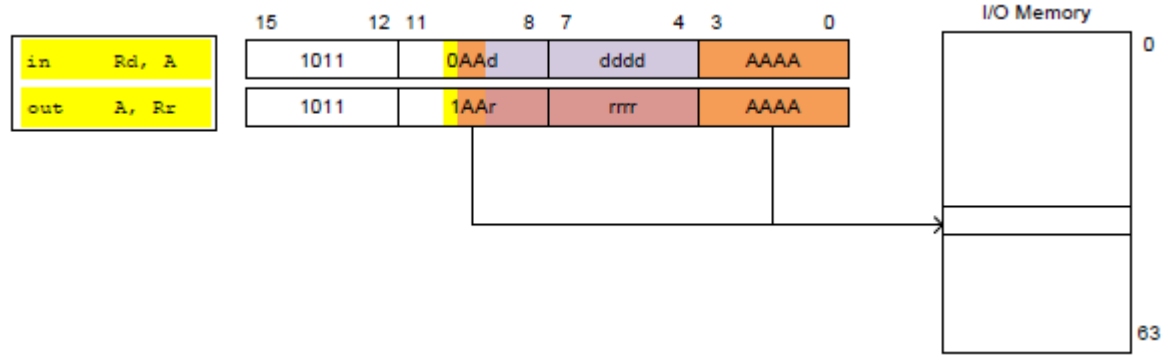


Команды установки флагов в регистре SREG

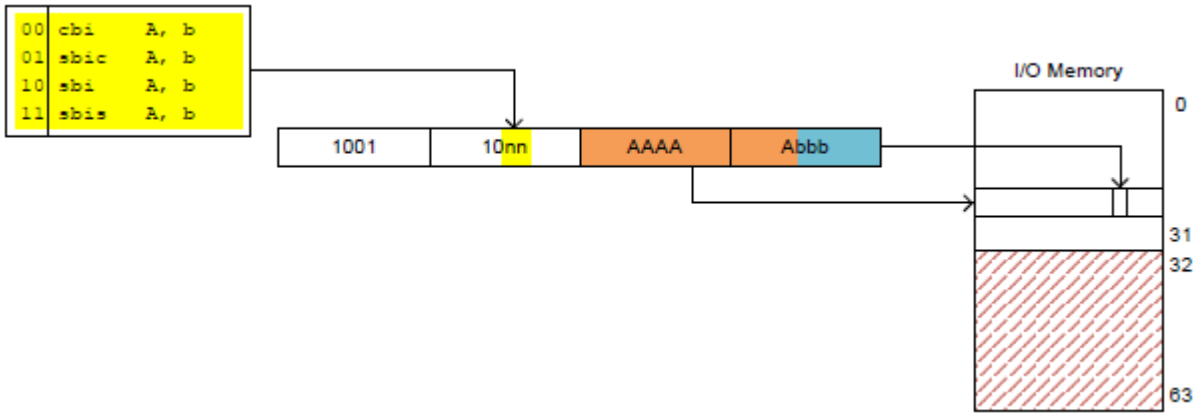


Команды конфигурирования портов ввода-вывода

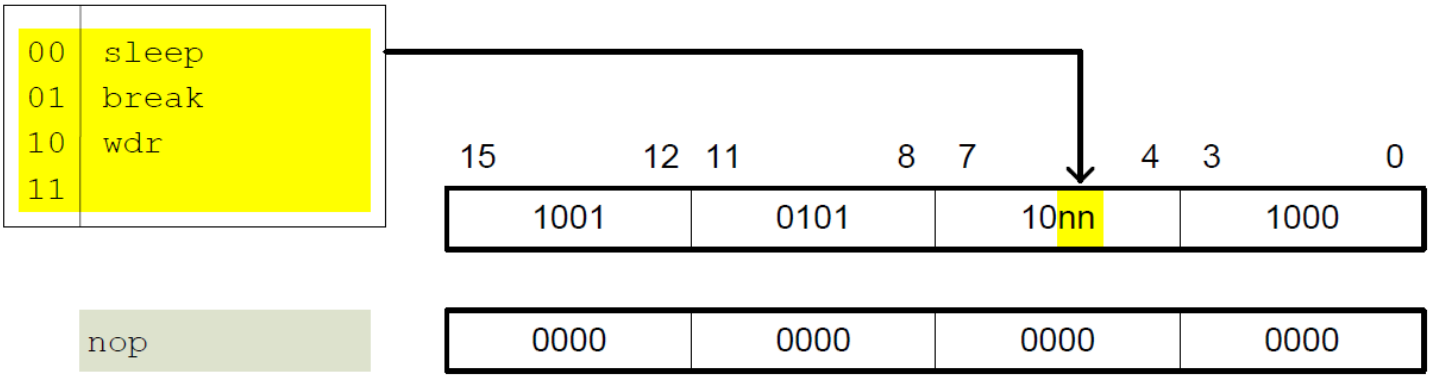
Команды установки/считывания всех разрядов портов ввода-вывода



Команды установки/считывания/переходов по отдельным разрядам портов ввода-вывода



Управляющие команды



Средства упрощения программирования на языке ассемблера

1. Переименование регистров:
.def TMP = R16
.def A0 = R20
.def A1 = R21

2. Сохранение констант:
.equ FREQ = 8000000

3. Макросы:

```
.macro outi ; Макрос для записи константы в порт, например, outi DDRD, 0xFC
    ldi TMP, @1
    out @0, TMP
.endm
```

Применение пользовательского разряда T в регистре SREG

Пользовательский разряд T регистра SREG позволяет программисту существенно упростить участки кода, оперирующие одноразрядной логической переменной. В МК Atmega32 существует всего 6 команд, предназначенные для работы с флагом T:

SET		Установка флага T	T ← 1

CLT		Очистка флага T	T ← 0
BST	Rr, b	Сохранить бит регистра Rr во флаг T регистра SREG	T ← Rr(b)
BLD	Rd, b	Считать флаг T регистра SREG в бит регистра Rd	Rd(b) ← T
BRTS	k, k∈[-64;63]	Переход при установленном флаге передачи (T=1)	if (T = 1) then PC ← PC + k + 1
BRTC	k, k∈[-64;63]	Переход при снятом флаге передачи (T=0)	if (T = 0) then PC ← PC + k + 1

Примеры использования:

1. Программа имеет два режима работы, в зависимости от которого изменяется логика работы основного цикла и прерываний

```
    _main:
CLT; установка первого режима
…
    _loop:
BRTS _loop_mode2; нужна 1 команда для определения режима
_loop_mode1: ; начало обработки первого режима программы
…
RJMP _loop
    _loop_mode2: ; начало обработки второго режима программы
…
RJMP loop

    _int0:
SET; переключение на второй режим работы
RETI

    _int1:
BLD TMP, 0
COM TMP ; переключение между режимами работы
BST TMP, 0
RETI

    _timer0_cmp:
BRTS _ timer0_mode2; нужна 1 команда для определения режима
_ timer0_mode1: ; начало обработки первого режима программы
…
RJMP _timer0_exit
    _ timer0_mode2: ; начало обработки второго режима программы
…
; RJMP _timer0_exit
    _timer0_exit:
RETI
```

2. Организация циклического сдвига

```
BST A0, 0 ; сохранение младшего бита во флаге T
LSR A0 ; логический сдвиг вправо
BLD A0, 7 ; заполнение 7 бита значением из флага T
```

Обработка многоразрядных данных в МК Atmega32

1. В МК Atmega32 существует ряд команд, предназначенных для обработки регистровых пар. В основном эти команды используются для

работы с десятиразрядными адресами в ОЗУ и имеют существенные ограничения:

ADIW	Rdl, K	dl∈{24,26,28,30}, K∈[0;63]	Сложение константы с регистровой парой	Rdh:Rdl ← Rdh:Rdl + K
SBIW	Rdl,K	dl∈{24,26,28,30}, K∈[0;63]	Вычитание константы из регистровой пары	Rdh:Rdl ← Rdh:Rdl - K
MOVW	Rd, Rr	d∈{0,2,…,30}, r∈{0,2,…,30}	Копирование регистровой пары	Rd+1:Rd ← Rr+1:Rr

2. Существуют ряд команд, учитывающих состояние флага переноса, что позволяет использовать их для обработки фрагментов многоразрядных чисел:

ADC	Rd, Rr	Сложение с учётом флага переноса	Rd ← Rd + Rr + C
SBC	Rd, Rr	Вычитание с учётом флага переноса	Rd ← Rd - Rr - C
SBCI	Rd, K, d∈[16;31]	Вычитание константы с учётом флага переноса	Rd ← Rd - K - C
ROL	Rd	Циклический сдвиг влево (через флаг переноса)	Rd(0)←C, Rd(n+1)← Rd(n), C←Rd(7)
ROR	Rd	Циклический сдвиг вправо (через флаг переноса)	Rd(7)←C, Rd(n)← Rd(n+1), C←Rd(0)
CPC	Rd,Rr	Сравнение с учётом флага переноса	Rd - Rr - C

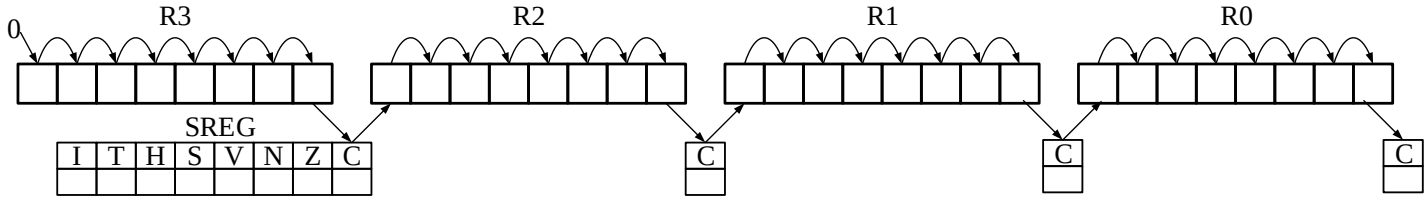
Примеры обработки многоразрядных чисел:

1. Сравнение двух 32-разрядных чисел:

```
CPI R0, R3
CPC R1, R4 ; сравнение с учётом результата предыдущего сравнения
CPC R2, R5
```

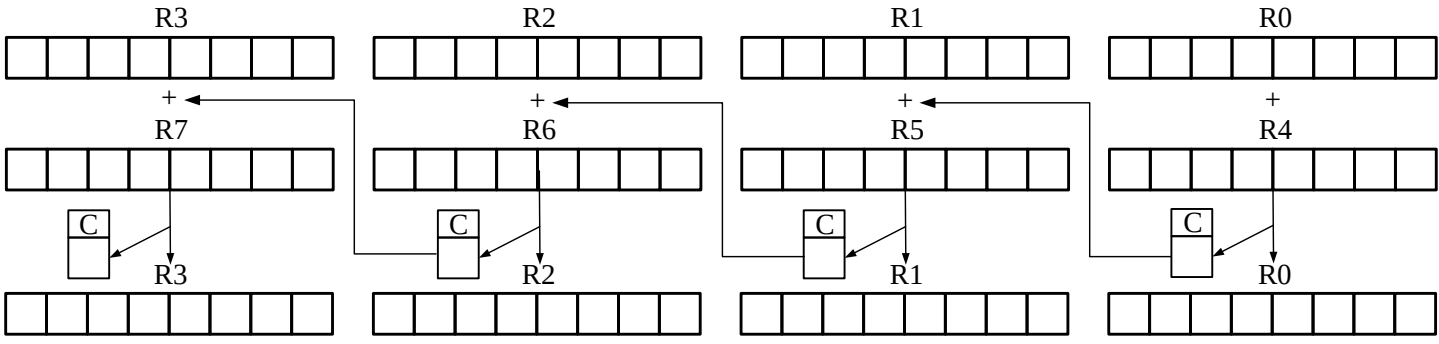
2. Арифметический сдвиг вправо 32-разрядного числа:

```
LSR R3
ROR R2 ; сдвиг с учётом младшего разряда старшего байта
ROR R1
ROR R0
```

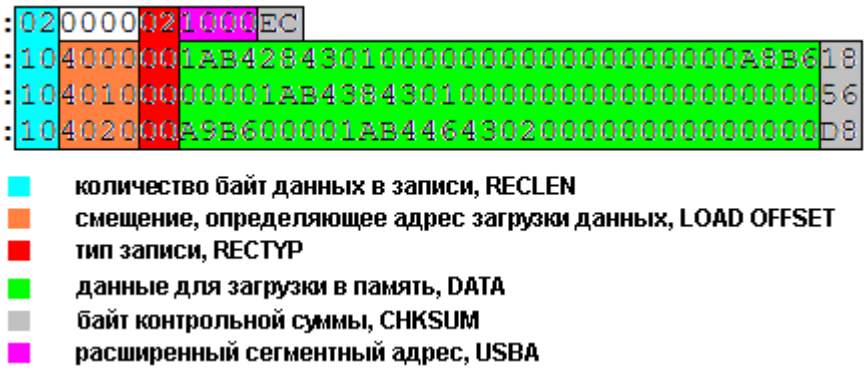


3. Сложение двух 32-разрядных чисел:

```
ADD R0, R4
ADC R1, R5 ; сложение с учётом переполнения при сложении младших байтов
ADC R2, R6
ADC R3, R7
```



Формат «.hex»-файла



Шестнадцатеричное представление двоичного файла закодировано с помощью цифробуквенных символов ASCII. Шестнадцатеричный объектный файл разбит на блоки записей (строки), каждая из которых содержит тип записи, длину, адрес загрузки в память и дополнительную контрольную сумму. Существует три основных типа записей:

- **Data Record**, запись для данных (8-, 16- или 32-bit форматы)
- **End of File Record**, запись для сигнала о конце файла (8-, 16- или 32-bit форматы)
- **Extended Segment Address Record**, запись для адреса расширенного сегмента (16- или 32-bit форматы)

Общий формат записи (General Record Format)

RECORD MARK ':'	RECLEN	LOAD OFFSET	RECTYP	INFO или DATA	CHKSUM
1 байт	1 байт	2 байта	1 байт	n байт	1 байт

Каждая запись в HEX-файле (строка) начинается с поля маркера начала записи **RECORD MARK**, содержащего ASCII-код 03AH, символ двоеточия ':'.

Следующее поле каждой записи **RECLEN**, которое задаёт количество байт полезной информации, которая содержится в записи (эти байты идут за полем RECTYP). Помните, что один байт данных представлен двумя символами ASCII. Максимальное значение для поля RECLEN является шестнадцатеричное 'FF', т. е. полезных данных в записи может быть от 0 до 255 байт.

Следующее поле в каждой записи **LOAD OFFSET**, которое указывает 16-битный начальный адрес загрузки байт данных, так что это поле используется только для записей данных (Data Record). В других типах

записей, где это поле не используется, оно должно быть закодировано четырьмя ASCII-символами нуля ('0000' или 030303030H).

Следующее поле в каждой записи **RECTYP**, которое обозначает тип этой записи. Поле RECTYP используется для интерпретации информации в записи. Вот как кодируются типы записей:

- '**00**' Data Record (запись, содержащая данные)
- '**01**' End of File Record (запись, сигнализирующая о конце файла)
- '**02**' Extended Segment Address Record (запись адреса расширенного сегмента)

Data Record (8-, 16- или 32-битный формат)

RECORD MARK ':'	RECLEN	LOAD OFFSET	RECTYPE '00'	DATA	CHKSUM
1 байт	1 байт	2 байта	1 байт	n байт	1 байт

Запись Data Record предоставляет набор шестнадцатеричных цифр, в которых находится ASCII-код байтов данных, которые содержатся в порции данных образа памяти. В отдельных полях записи имеется следующее содержимое:

- RECORD MARK – Это поле содержит байт 03AH, символ двоеточия, закодированный шестнадцатеричным символом ASCII (':').
- RECLEN – Это поле содержит две шестнадцатеричные цифры ASCII, которые задают количество байт данных, находящихся в записи. Максимальное значение равно 'FF' или 04646H (что соответствует десятичному значению 255).
- LOAD OFFSET – Это поле содержит четыре шестнадцатеричные цифры ASCII, представляющие смещение от LBA (см. раздел Extended Linear Address Record) или SBA (см. раздел Extended Segment Address Record), что задает адрес, куда будет помещен первый байт записи.
- RECTYP – Это поле содержит байты 03030H, кодирующие шестнадцатеричными символами ASCII '00', что задает тип записи Data Record.
- DATA – Это поле содержит пары шестнадцатеричных цифр ASCII, где каждая пара кодирует один байт данных.
- CHKSUM – Это поле содержит контрольную сумму полей RECLEN, LOAD OFFSET, RECTYP и DATA.

Пример расчёта времени выполнения фрагмента кода

```
delay:
        LDI     R17, 198;  y = 198
        LDI     R16, 102;  x = 102
delay_sub:
        DEC     R16
        BRNE    delay_sub      } Внутренний цикл
        DEC     R17
        BRNE    delay_sub      } Внешний цикл
        NOP
```

Команда	Число тактов
LDI	1 - загрузка
DEC	1 - декремент
BRNE	1 - сравнение 1/0 - переход
NOP	1 - простой

Флаг-бит Z устанавливается в 1 при нулевом результате операции
Команда условного перехода BRNE выполняет переход, если Z = 0
Число тактов до первого выхода из внутреннего цикла:

$N_{вн.1} = (1_{DEC} + 2_{BRNE}) \times (x - 1) + 1_{DEC} + 1_{BRNE} = 3 \times x - 1$

Число тактов до последующих выходов из внутреннего цикла:

$N_{вн.x} = (1_{DEC} + 2_{BRNE}) \times (2^8 - 1) + 1_{DEC} + 1_{BRNE} = 3 \times 2^8 - 1$

Общее число тактов:

$N_{общ} = 1_{LDI} \times 2 + (N_{вн.1} + 1_{DEC} + 2_{BRNE}) + (N_{вн.x} + 1_{DEC} + 2_{BRNE}) \times (y - 2) + (N_{вн.x} + 1_{DEC} + 1_{BRNE}) + 1_{NOP} =$

$= 3 \times x + 770 \times (y - 1) + 4 = 152\,000$

Время выполнения: $t = N / T = 152\,000 / 8\,000\,000 = 0{,}019\ с = 19\ мс$

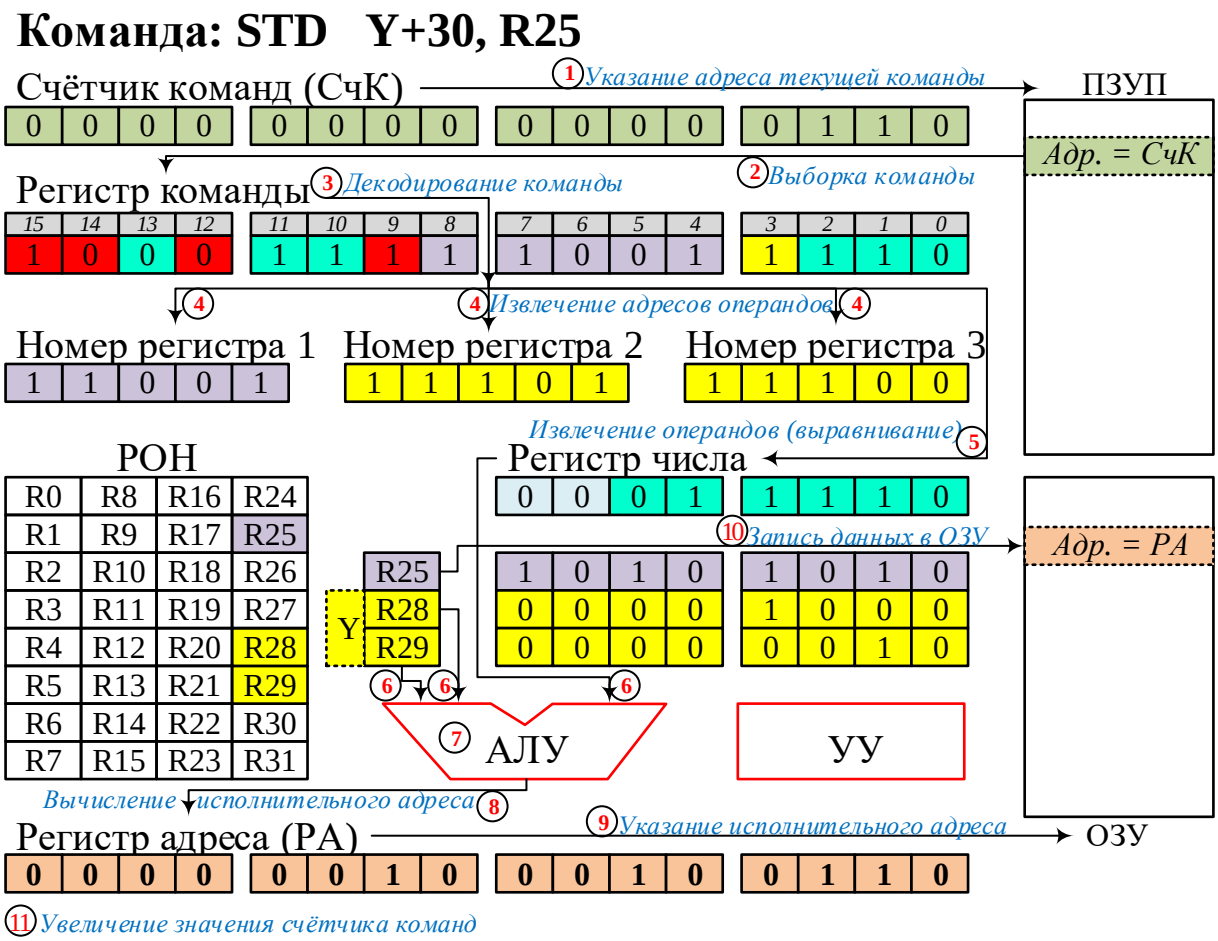
Командный цикл процессора на архитектуре ATmega32

При выполнении каждой команды процессор выполняет ряд действий, некоторый из которых являются обязательными для всех команд, а остальные зависят от выполняемой команды. В следующем перечне

действий зелёным выделены обязательные этапы командного цикла, а красным выделены действия, выполняемые исключительно в командах работы с ОЗУ (LD* и ST*), а также при обращении к расположенному в ОЗУ стеком (PUSH, POP, CALL*, RET*):

1. **Передача флэш-памяти программ адреса текущей команды (из счётчика команд)**
2. **Увеличение счётчика команд на 1**
3. **Извлечение команды из флэш-памяти программ и запись в регистр команды**
4. **Декодирование команды**
5. Извлечение адресов операндов (**прямая адресация**), возможно дополнение предопределённых старших разрядов
6. Извлечение операндов (**непосредственная адресация**), возможно выравнивание
7. Извлечение байта/бита из порта ввода-вывода
8. Передача операндов в АЛУ
9. Выполнение операции
10. **Извлечение байта из ОЗУ (LD*, POP, RET*)**
11. Обновление статусов (флаги в SREG и т.п.)
12. Запись результата в РОН
13. Запись результата в порт ввода-вывода
14. **Запись результата в ОЗУ (ST*, PUSH, CALL*)**
15. Запись результата в счётчик команд
16. **Переход к пункту 1**

Пример порядка выполнения команды STD в МК ATmega32



Этапы выполнения команд:

1. Передача памяти программ (ПЗУП) адреса текущей команды
2. Извлечение команды из ПЗУП и запись в регистр команды
3. Декодирование команды – определение типа операции и формата
4. Извлечение адресов операндов (номеров регистров общего назначения (РОН))
5. Извлечение операндов из команды (непосредственная адресация) и выравнивание данных (в данном случае расширение знакового операнда с 6 до 8 разрядов)
6. Передача операндов в сумматор
7. Выполнение операции суммирования
8. Передача результата операции суммирования в регистр адреса
9. Передача исполнительного адреса в ОЗУ
10. Запись данных из R25 в ОЗУ
11. Увеличение значения счётчика команд

Для других команд могут встречаться этапы считывания или записи флагов регистра SREG, чтение или запись из портов ввода-вывода и другие, неуказанные в примере выше.

Порядок выполнения работы

1. В базовом примере (см. ниже) заменить фрагмент кода, выделенный зелёным, на фрагмент из своего варианта.

2. Создать новый проект в Atmel Studio на языке ассемблера, разместить исходный код из п.1 в файле проекта. Сформировать «.lss»-файл и «.hex»-файл. Записать «.hex»-файл в контроллер с помощью программы AVRFlash.

3. Изучить архитектуру и систему команд микроконтроллера ATmega32. Разобраться в алгоритме работы текущей программы. Убедиться в правильности работы программы.

4. Определить зависимость количества тактов, за которое выполняется заменённый блок кода, от констант x и y.

5. Вычислить значения констант x и y при которых блок кода «delay» будет выполняться ровно 0,01 секунды (при тактовой частоте 8 МГц). При невозможности обеспечения точной величины задержки необходимо дополнить блок кода соответствующим количеством команд NOP (на месте закомментированной команды NOP в базовом примере может быть добавлено до 4 команд NOP).

6. Изучить сформированный «.lss»-файл, выписать адреса всех меток программы, перечислить используемые форматы команд в части состава и размерности операндов.

7. Изучить структуру сформированного «.hex»-файла, определить количество записей в файле и количество машинных слов программы.

8. Взять команду ассемблера в соответствии с вариантом и описать порядок её выполнения внутри центрального процессора: определить этапы командного цикла, задействованные узлы МК, состав пересылаемых данных и управляющих сигналов.

9. Укажите мнемонику команд и их аргументы в виде имён регистров общего назначения, имён портов ввода-вывода, имён разрядов в портах ввода-вывода и констант в десятичной системе счисления для машинных слов, указанных в варианте. Поместите полученные команды с аргументами в конец программы из п.1, сформируйте «.lss»-файл и найдите в нём соответствующие строки, сравните машинные слова с исходным заданием.

Базовый пример

```
.def TMP = R20

.org $000
   JMP reset ; Указатель на начало программы

; Функция паузы
delay: ; задержка 120 мс
   LDI  R31, 5
   LDI  R30, 223
   LDI  R29, 188
delay_sub:
   DEC  R29
   BRNE delay_sub
   DEC  R30
   BRNE delay_sub
   DEC  R31
   BRNE delay_sub
   NOP
   NOP
   RET

; Начальная настройка
reset:
; настройка исходных значений
   LDI  TMP, 0x01;
   MOV  R0, TMP
   CLR  TMP;
   MOV  R1, TMP
   MOV  R2, TMP
   MOV  R3, TMP
; настройка портов ввода-вывода
   SER  TMP ; 0xFF
   OUT  DDRA, TMP ; Вывод
   OUT  DDRB, TMP ; Вывод
   OUT  DDRC, TMP ; Вывод
   OUT  DDRD, TMP ; Вывод
; Установка вершины стека в конец ОЗУ
   LDI  TMP, HIGH(RAMEND) ; Старшие разряды адреса
   OUT  SPH, TMP
   LDI  TMP, LOW(RAMEND) ; Младшие разряды адреса
   OUT  SPL, TMP

; Основной цикл
loop:
; Циклический сдвиг 32-разрядного числа R0-R3
   BST R0, 0 ; сохранение младшего бита во флаге T
   LSR R3 ; логический сдвиг вправо
   ROR R2 ; циклический сдвиг вправо
   ROR R1 ; циклический сдвиг вправо
   ROR R0 ; циклический сдвиг вправо
   BLD R3, 7 ; заполнение 7 бита значением из флага T
```

```
; Вывод 32-разрядного числа R0-R3 на порты PORTA-PORTD
   OUT PORTA, R0
   OUT PORTB, R1
   OUT PORTC, R2
   OUT PORTD, R3
; Пауза
   CALL delay;
; Возврат в начало основного цикла
   RJMP loop ;
```

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Комментированный листинг программы на языке ассемблера.
4. Алгоритм выполнения задействованных в программе команд ассемблера.
5. Ответы на контрольные вопросы.

Контрольные вопросы

1. Перечислите основные узлы микроконтроллера ATmega32 и укажите их назначение.
2. Укажите, в чём проявляются признаки RISC-архитектуры в микроконтроллере ATmega32. В чём преимущества и недостатки приведённых особенностей?
3. От чего зависит время выполнения команд CPSE и BRHS? Приведите примеры кода (до 3-10 команд каждый), приводящие к различной продолжительности выполнения указанных команд.
4. Приведите пример выполнения циклического сдвига вправо 8-разрядного числа (с переходом младшего разряда в старший) без использования флага T. Приведите пример из трёх машинных команд, обеспечивающих сложение 24-разрядного числа с 24-разрядной константой.
5. Обоснуйте, чем вызваны ограничения допустимых значений номеров регистров и диапазонов констант в некоторых командах микроконтроллера ATmega32?

Варианты заданий

Номер варианта	Фрагмент кода (п. 1)	Команда (п. 8)	Машинные слова (п. 9)

1а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	DEC R29
	BRNE delay_sub
	NOP
	DEC R30
	BRNE delay_sub
	RET
		LD R30, X+	935f
			6018

1б		OUT DDRD, R5	b71f ef6f
1в		CBR R20, 100	9ae1 cffe

2а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	DEC R29
	NOP
	BRNE delay_sub
	DEC R30
	NOP
	BRNE delay_sub
	RET
		SBRS R6, 3	9309
			9400

2б		IN PINA, R4	e0ce 91cf
2в		ADIW R30, 10	952a 9550

3а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	INC R29
	BRNE delay_sub
	NOP
	DEC R30
	BRNE delay_sub
	RET
		SUB R21, R1	bb4a
			ef4f

3б		SBR R30, 76	2d1d fd41
3в		PUSH R0	e765 957a

4а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	INC R29
	BRNE delay_sub
	NOP
		SBIW R30, 63	f446
			9536

4б		CPSE R0, R1	8118 fd31
4в		LD R30, Y	1334 b7ff

	INC R30 BRNE delay_sub RET		
5а	delay: LDI R30, 200; y	SBRC R17, 3	0f44 9518
5б	LDI R29, 153; x delay_sub:	MULSU R19, R20	b7ff 9518
5в	DEC R29 NOP BRNE delay_sub DEC R30 BRNE delay_sub RET	SBI DDRC, 3	fd40 cff5
6а	delay: LDI R30, 10; y	SBIS PORTA, 0	9468 f3e1
6б	LDI R29, 153; x delay_sub:	MULS R21, R20	f3fe 9478
6в	INC R29 NOP BRNE delay_sub NOP INC R30 BRNE delay_sub RET	LD R5, Z+	bf1c ec10
7а	delay: LDI R30, 200; y	ADD R12, R8	cffe 99e1
7б	LDI R29, 153; x delay_sub:	SBI PORTD, 7	f039 bb3b
7в	INC R29 NOP BRNE delay_sub NOP DEC R30 BRNE delay_sub RET	ADC R15, R16	Bfff 94e8
8а	delay: LDI R30, 200; y	SBIC DDRC, 1	9536 f446
8б	LDI R29, 153; x delay_sub:	BST R3, 7	8118 b7ff
8в	INC R29 BRNE delay_sub DEC R30	BRLT -20	9309 fd41

	NOP BRNE delay_sub RET		

9а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	NOP
	DEC R29
	NOP
	BRNE delay_sub
	INC R30
	BRNE delay_sub
	RET
		CLR R7	1334
			fd31

9б		ANDI R18,100	9550 e765
9в		CPSE R6, R7	2d1d bb4a

10а	delay:
	LDI R30, 200; y
	LDI R29, 153; x
	delay_sub:
	INC R29
	NOP
	BRNE delay_sub
	DEC R30
	NOP
	NOP
	BRNE delay_sub
	RET
		CBI ADMUX, 5	9400
			ef4f

10б		MUL R0, R2	952a 6018
10в		ROL R7	91cf e0ce

Лабораторная работа 2

ОРГАНИЗАЦИЯ ЦИФРОВОГО ВВОДА-ВЫВОДА

Цель работы

Изучение основ работы с цифровыми портами ввода-вывода микроконтроллера ATmega32. Получение практических навыков по обработке внешних прерываний и организации ввода-вывода с помощью механизма прерываний.

Теоретические сведения

Порты ввода-вывода

Связь микроконтроллера с внешними устройствами осуществляется через порты ввода-вывода. Каждый разряд порта соответствует одному из выводов микросхемы. В зависимости от модели микроконтроллера может быть от одного до семи восьмиразрядных портов (иногда присутствуют неполные порты, имеющие менее восьми разрядов), именующихся латинскими буквами от A до G. Упрощённая схема одного вывода представлена на рисунке 3. Физическое расположение выводов можно найти в спецификации микроконтроллера.

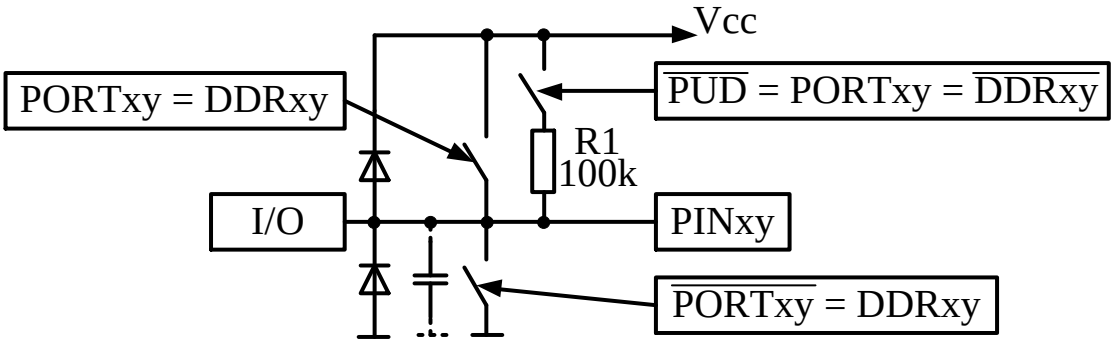


Рис. 1. Схема порта ввода-вывода

Управляется каждый порт тремя специальными регистрами – PORTx, DDRx и PINx, где под «x» подразумевается конкретная буква – имя порта. Отдельные разряды регистров также имеют свои имена. Разряды регистра PORTx именуются Pxi, где «i» – номер соответствующего разряда. Например, разряды порта A именуются PA0, PA1, …, PA7. Аналогично,

разряды порта DDRx именуются DDxi (DDA0, ..., DDA7 для порта A), а разряды регистра PINx именуются PINxi (PINA0, ..., PINA7).

Каждый контакт микросхемы устроен так, что может работать как на ввод, так и на вывод информации, а для переключения режима (ввод/вывод) служит регистр DDRx. Каждый разряд этого регистра управляет соответствующим разрядом порта – если разряд равен нулю, то разряд порта работает как вход. Если разряд регистра равен единице, то разряд порта работает как выход. Таким образом, все выводы микроконтроллера настраиваются независимо друг от друга.

Каждый вывод имеет в своём составе диод, защищающий от превышения напряжения. Изображённый пунктиром конденсатор обозначает паразитарную ёмкость вывода – несмотря на её маленькую величину, она всё же присутствует, и иногда её приходится учитывать. Всё управление выводом осуществляется ключами управления через регистры порта. Для простоты изложения на схеме ключи обозначены в виде переключателей, хотя в реальности это полевые транзисторы.

Чтобы выдать нужный логический уровень на вывод микросхемы, нужно записать в соответствующий разряд регистра DDRx единицу, а затем записать бит данных в соответствующий разряд регистра PORTx. Логический уровень будет присутствовать на выводе порта до тех пор, пока линия не переключится на ввод, либо не изменится значение регистра PORTx.

Для того чтобы прочитать информацию с внешнего вывода микроконтроллера, необходимо сначала перевести нужный разряд порта в режим ввода. То есть записать в соответствующий разряд регистра DDRx ноль. Только после этого на данный вывод микроконтроллера можно подавать цифровой сигнал от внешнего устройства. Далее микроконтроллер просто читает байт из регистра PINx. Содержимое соответствующего бита прочитанного байта будет соответствовать сигналу на внешнем выводе порта.

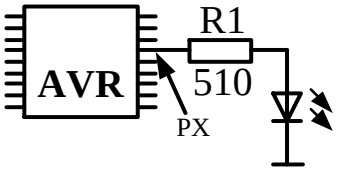
Каждый вывод дополнительно оснащён подтягивающим (pull-up) резистором. Подтягивающие резисторы не имеют независимого управления и могут быть подключены либо отключены одновременно установкой/сбросом бита PUD (PullUp Disable) регистра SFIOR. Логика управления является инверсной, т.е. установка бита отключает резисторы. По умолчанию бит равен нулю, и подтягивающие резисторы включены.

Пример подключения внешних устройств

Исследовать работу портов ввода-вывода можно при помощи светодиодов и контактных кнопок, подключённых к выводам микроконтроллера. Напишем программу, которая будет при нажатии на подключённую к микроконтроллеру кнопку зажигать/гасить светодиод.

Для начала разберём схему подключения. Поскольку логической единице соответствует напряжение 5 В, а рабочее напряжение светодиодов составляет около 3 В, то подключать светодиод напрямую нельзя – это приведёт к его сгоранию. Для правильного подключения нужно использовать сопротивление. Подключить светодиод можно двумя способами.

Зажигание логической единицы



Зажигание логического нуля

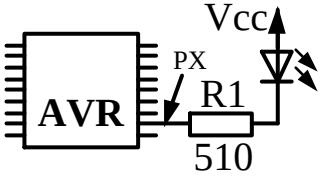


Рис. 2. Различные способы использования порта

В первом случае для включения светодиода нужно сконфигурировать порт на вывод, и записать в него единицу. Во втором случае нужно сконфигурировать порт на ввод, и записать в него ноль. Мы воспользуемся первым способом, как более прозрачным с точки зрения логики.

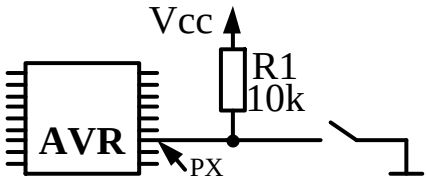


Рис. 3. Вариант подключения кнопки

Разберём чуть подробнее схему подключения кнопки. Для корректной работы кнопки регистр ввода-вывода должен быть сконфигурирован на ввод. На рисунке 5 изображён подтягивающий резистор *R1*. Подтягивающие резисторы уже встроены в микроконтроллер, и на схеме резистор *R1* приведён только для наглядности изложения. Видно, что когда кнопка отпущена, с входа микроконтроллера будет считываться логическая единица, т. к. вход подтянут резистором к линии питания. Когда кнопка нажата, то линия питания соединяется с землёй через резистор, при этом из регистра PINx будет считан логический ноль.

Подсистема прерываний

Все микроконтроллеры AVR имеют многоуровневую систему прерываний. Прерывание прекращает нормальный ход программы для выполнения приоритетной задачи, определяемой внутренним или внешним событием.

Для каждого такого события разрабатывается отдельная программа, которую называют подпрограммой обработки запроса на прерывание (для краткости – подпрограммой прерывания), и размещается в памяти программ.

У каждого периферийного устройства, что входит в состав AVR микроконтроллеров, есть как минимум один источник прерывания (Interrupt source). Ко всем этим прерываниям следует причислить и прерывание сброса – Reset Interrupt, предназначение которого отличается от всех остальных.

За каждым прерыванием, строго закреплён вектор (ссылка) указывающий на процедуру обработки прерывания (Interrupt service routine). Все векторы прерываний, располагаются в самом начале памяти программ и вместе формируют «таблицу векторов прерываний» (Interrupt vectors table).

Каждому прерыванию соответствует определённый «бит активации прерывания» (Interrupt Enable bit). Таким образом, чтобы использовать определённое прерывание, следует записать в его «бит активации прерывания» – логическую единицу. Далее, независимо от того активированы ли определённые прерывания, микроконтроллер не начнёт обработку этих прерываний, пока в «бит всеобщего разрешения прерываний» (Global Interrupt Enable bit в регистре состояния *SREG*) не будет записана логическая единица. Также, чтобы запретить все прерывания (на неопределённое время), в бит всеобщего разрешения прерываний следует записать – логический ноль.

В таблице 1 приведён перечень источников прерываний для МК ATmega32.

Таблица1 — Источники прерываний в МК ATmega32

№	Адрес	Источник	Описание
1	0x000	RESET	Сигнал сброса
2	0x002	INT0	Внешний запрос на прерывание по входу INT0

3	0x004	INT1	Внешний запрос на прерывание по входу INT1
4	0x006	INT2	Внешний запрос на прерывание по входу INT2
5	0x008	TIMER2_COMP	Совпадение с регистром сравнения таймера T/C2
6	0x00A	TIMER2_OVF	Переполнение счётчика T/C2
7	0x00C	TIMER1_CAPT	Захват по таймеру T/C1
8	0x00E	TIMER1_COMPA	Совпадение с регистром сравнения A таймера T/C1
9	0x010	TIMER1_COMPB	Совпадение с регистром сравнения B таймера T/C1
10	0x012	TIMER1_OVF	Переполнение счётчика T/C1
11	0x014	TIMER0_COMP	Совпадение с регистром сравнения таймера T/C0
12	0x016	TIMER0_OVF	Переполнение счётчика T/C0
13	0x018	SPI_STC	Передача данных по интерфейсу SPI завершена
14	0x01A	UART_RXC	Приём данных приёмопередатчиком UART завершён
15	0x01C	UART_UDRE	Регистр данных UART пуст
16	0x01E	UART_TXC	Передача данных приёмопередатчиком UART завершена
17	0x020	ADC	Завершено преобразование АЦП
18	0x022	EE_RDY	EEPROM готов
19	0x024	ANA_COMP	Прерывание от аналогового компаратора
20	0x026	TWI	Прерывание от интерфейса I2C
21	0x028	SPM_RDY	Запись программной памяти (Flash) готова

Управление внешними прерываниями

За управление внешними прерываниями в ATmega32 отвечают четыре регистра:

– GICR (он же GIMSK) – запрет/разрешение прерываний по сигналам на входах INT0, INT1;

– MCUCR – выбор условия срабатывания прерываний int0 и int1;

– GIFR – управление внешними прерываниями;

7	6	5	4	3	2	1	0
INT1	INT0	-	-	-	-	-	-

Рис. 4. Регистр GICR

INTx=1: прерывания по сигналу на входы INTx разрешены.

7	6	5	4	3	2	1	0
SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00

Рис. 5. Регистр MCUCR

ISC01 и ISC00 – управление прерыванием int0:

ISC01	ISC00	Описание
0	0	Низкий уровень напряжение вызывает прерывание
0	1	Любое изменение логического сигнала вызывает прерывание
1	0	Перепад 1/0 вызывает прерывание
1	1	Перепад 0/1 вызывает прерывание

Аналогично биты ISC11 и ISC10 управляют прерыванием int1.

7	6	5	4	3	2	1	0
INTF1	INTF0	-	-	-	-	-	-

Рис. 6. Регистр GIFR

INTFx=1: произошло прерывание на входе INTx. При входе в подпрограмму обработки прерывания INTFx автоматически сбрасывается в состояние логического нуля.

При входе в подпрограмму обработки прерывания соответствующий прерыванию флаг регистра GIFR автоматически сбрасывается в состояние логический ноль.

Прерывания работают только тогда, когда в регистре состояния SREG разрешены общие прерывания (бит 7 равен единице). В случае наступления прерывания этот бит автоматически сбрасывается в ноль, блокируя выполнение последующих прерываний.

Примеры решения задач

Считывание нескольких нажатых кнопок

Учитывая высокую частоту работу микроконтроллера, при вводе многоразрядного числа с помощью кнопок, подключённых к отдельным разрядам портов ввода-вывода, необходимо обеспечить временную задержку между зажатием первой кнопки и вводом необходимой комбинации кнопок. Кроме этого, необходимо обеспечить защиту от повторного выполнения фрагмента кода, обеспечивающего ввод числа, – проще всего установить цикл, ожидающий пока все кнопки не будут отпущены, перед выходом из фрагмента кода.

Рис. 7. Блок-схема считывания многоразрядного числа

Пример кода на ассемблере:

```
.def NULL = R16
.def TMP  = R17

read_number:
   IN   TMP, PINB
   CP   TMP, NULL ; PINB = 0?
   BREQ read_number; если PINB = 0 возврат в начало функции
   CALL delay ; ожидание нажатия комбинации кнопок
   IN   TMP, PINB ; чтение комбинации кнопок
   CALL operation ; выполнение операции
stop_reading: ; обеспечение однократного ввода
   IN   TMP, PINB
   CP   TMP, NULL ; PINB = 0?
   BRNE stop_reading ; ожиданіе условія PINB == 0
   RET
```

Пример кода на си:

```
uint_8t get_num() {
  uint_8t Num;
  while(!PINA); // ожидания нажатия каких-либо кнопок на порту A
  delay_ms(120);// пауза для завершения ввода сочетания кнопок
  Num = PINA;   // считывание сочетания кнопок
// обработка введённого числа
  while(PINA);  // ожидание завершения ввода (обеспечение однократной
обработки введённого значения)
}
```

Пример использования обработчика прерывания int0

```
.org $000
   JMP RESET            ; Указатель на начало программы
.org INT0addr
   JMP EXT_INT0   ; Указатель на обработчик прерывания int0
.org INT1addr
   JMP EXT_INT1   ; Указатель на обработчик прерывания int1

RESET:              ; Начало основной программы
      LDI R20, HIGH(RAMEND) ; Старшие разряды адреса
      OUT SPH, R20            ; Установка вершины стека в конец ОЗУ
      LDI R20, LOW(RAMEND)  ; Младшие разряды адреса
      OUT SPL, R20            ; Установка вершины стека в конец ОЗУ
      SER R16
      OUT DDRA, R20           ; Настройка PORTA на вывод
      OUT DDRB, R20           ; Настройка PORTB на вывод
      LDI R16, 0x0F
      OUT MCUCR, R16    ; Настройка прерываний int0 и int1 на условие 0/1
      LDI R16, 0xC0
      OUT GICR, R16           ; Разрешение прерываний int0 и int1
      OUT GIFR, R16           ; Предотвращение срабатывания int0 и int1 при
включении прерываний
      SEI               ; Включение прерываний

LOOP:
      NOP
      ; ...
      ; здесь могут быть операции с регистром R16, например:
      ; DEC R16 ; <- если поступит прерывание и в обработчике будет перезаписан
R16 (1)
      ; а также условные переходы, например:
      ; CPI R16, 0xef ; <- если поступит прерывание и в обработчике будет
перезаписан SREG (2)
      ; BRNE LABEL1 ; <- при выполнении условий (1) или (2) переход (не) будет
выполнен ошибочно
      ; ...
      RJMP LOOP   ; Бесконечный цикл

EXT_INT0:         ; Обработчик прерывания int0
      PUSH R16    ; Сохранение текущего значения R16 в стеке
      IN R16, SREG
      PUSH R16          ; Сохранение текущего значения SREG в стеке
      IN R16, PORTA           ; (!) изменяется значение R16
      COM R16           ; (!!!) изменяются значения флагов в SREG
      OUT PORTA, R16    ; Инверсия значения на PORTA
      POP R16
      OUT SREG, R16           ; Восстановление значения SREG из стека
      POP R16           ; Восстановление значения R16 из стека
      RETI              ; Возврат из обработчика прерываний и разрешение
прерываний

EXT_INT1:               ; Обработчик прерывания int1
      PUSH R16          ; Сохранение текущего значения R16 в стеке
      IN R16, SREG
      PUSH R16          ; Сохранение текущего значения SREG в стеке
      IN R16, PORTB           ; (!) изменяется значение R16
```

```
        COM R16             ; (!!!) изменяются значения флагов в SREG
        OUT PORTB, R16      ; Инверсия значения на PORTB
        POP R16
        OUT SREG, R16           ; Восстановление значения SREG из стека
        POP R16             ; Восстановление значения R16 из стека
        RETI                ;  Возврат  из  обработчика  прерываний  и  разрешение
прерываний
```

Пример алгоритма для калькулятора

Задача: реализовать простейший восьмиразрядный калькулятор. Регистр PORTA отображает результат предыдущей операции или первый введённый операнд. Регистр PORTB используется для ввода восьмиразрядного знакового операнда в двоичной системе счисления (необходима временная задержка, позволяющая зажать несколько кнопок одновременно). Разряды регистра PORTC отвечают за указание математической операции (PC1 – логическое сложение, PC2 – арифметическое вычитание) или обнуление первого операнда (PORTA) – PC0 и вывод флагов (переполнение – PC7 и отрицательный результат – PC6). Ввод данных должен осуществляться в следующей последовательности: ввод первого операнда, указание действия, ввод второго операнда. После выполнения первой операции в качестве первого операнда может выступать результат предыдущей операции. Ввод второго операнда до указания действия приводит к замене первого операнда. Повторное указание действия перед вводом второго операнда приводит к замене выполняемого действия.

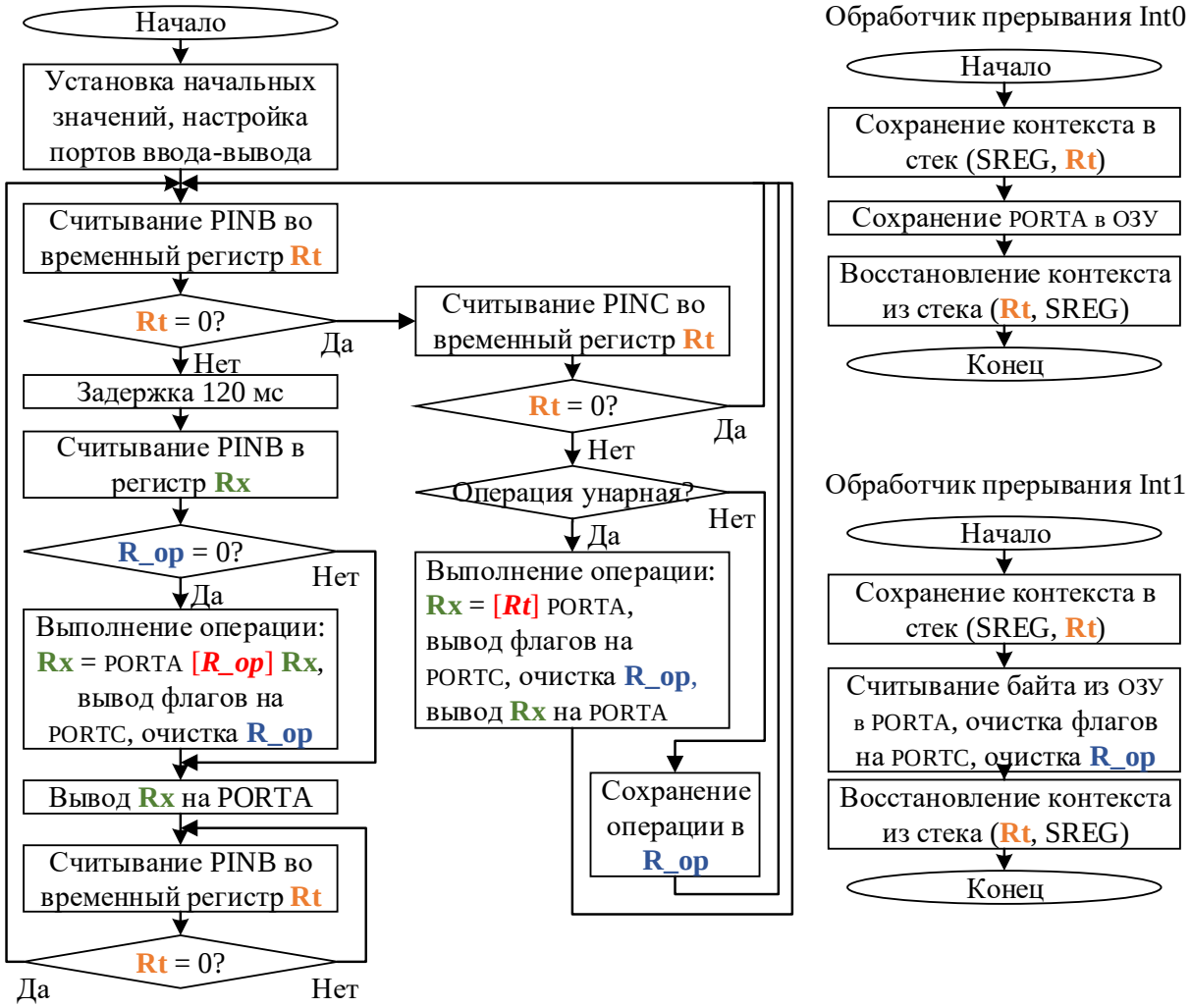


Рис. 8. Блок-схема алгоритма работы калькулятора

```
.def NULL = R16
.def TMP  = R17
.def TMP2 = R18
.def OP   = R19
.def FLAG = R20

reset:
; настройка исходных значений
   CLR  NULL ; 0x00
   CLR  OP   ; 0x00
; настройка портов ввода-вывода
   SER  TMP ; 0xFF
   OUT  DDRA, TMP ; Вывод
   CLR  TMP ; 0x00
   OUT  DDRB, TMP ; Ввод
   LDI  TMP, 0xF8
   OUT  DDRC, TMP ; 7,6 - вывод, 2,1,0 - ввод
; Установка вершины стека в конец ОЗУ
   LDI  TMP, HIGH(RAMEND) ; Старшие разряды адреса
   OUT  SPH, TMP
   LDI  TMP, LOW(RAMEND) ; Младшие разряды адреса
   OUT  SPL, TMP

read_number: ; считывание операнда
   IN   TMP, PINB
   CP   TMP, NULL ; PINB = 0?
   BREQ read_operation ; если PINB = 0 считывание операции
   CALL delay ; ожидание нажатия комбинации кнопок
   IN   TMP, PINB ; чтение комбинации кнопок
   CPSE OP, NULL ; если введена операция
   CALL operation ; выполнение операции
   OUT  PORTA, TMP ; вывод операнда или результата операции
stop_reading: ; обеспечение однократного ввода
   IN   TMP, PINB
   CP   TMP, NULL ; PINB = 0?
   BRNE stop_reading ; ожидание условия PINB == 0
   RJMP read_operation

read_operation: ; считывание операции
   IN   TMP, PINC
   CP   TMP, NULL ; PINC = 0?
   BREQ read_number ; если PINC = 0 считывание операнда
   SBRC TMP, 0 ; если операция не равна очистке: PC.0 - очистка
   RJMP op_null ; выполнение операции очистки
   MOV  OP, TMP ; сохранение операции
   RJMP read_number

operation: ; начало бинарной операции
   IN   TMP2, PORTA ; считывание первого операнда из PORTA
   SBRC OP, 1 ; определение операции: PC.1 - OR, PC.2 - SUB
   RJMP op_or ; если PC1 = 1 выполнение операции OR
   RJMP op_sub ; если PC1 = 0 выполнение операции SUB

op_null: ; операция очистки
   OUT  PORTA, NULL ; очистка операнда
   MOV  FLAG, NULL ; очистка флагов
   CALL end_of_operation
   RJMP read_number

op_or: ; операция OR
   OR   TMP, TMP2 ; операция OR
   IN   FLAG, SREG ; сохранение флагов
   RJMP end_of_operation

op_sub: ; операция SUB
   SUB  TMP2, TMP ; операция SUB
   MOV  TMP, TMP2 ; перенос результата
   IN   FLAG, SREG ; сохранение флагов
   RJMP end_of_operation
```

```
end_of_operation: ; завершение операции
   SWAP FLAG ; замена тетрад: PC.7 = SREG.3 (V), PC.6 = SREG.2 (N)
   ANDI FLAG, 0xC0 ; очистка других флагов
   OUT  PORTC, FLAG ; вывод флагов
   MOV  OP, NULL ; очистка операции
   RET

delay: ; задержка 120 мс
   LDI  R31, 5
   LDI  R30, 223
   LDI  R29, 188
delay_sub:
   DEC  R29
   BRNE delay_sub
   DEC  R30
   BRNE delay_sub
   DEC  R31
   BRNE delay_sub
   NOP
   NOP
   RET
```

Представление чисел со знаком

1. Прямой код:

Старший бит кодирует знак

Остальные биты совпадают у положительного и отрицательного числа

2. Обратный код (дополнение до 1):

Старший бит кодирует знак

a + (-a) = 2n-1, где n – количество разрядов числа

3. Дополнительный код (дополнение до 2):

Старший бит кодирует знак

a + (-a) = 2n, где n – количество разрядов числа

Примеры:

Число		Прямой код	Обратный код	Дополнительный код
23	10111	00010111	00010111	00010111
127	1111111	01111111	01111111	01111111
1	1	00000001	00000001	00000001
-1	-1	10000001	11111110	11111111
-17	-10001	10010001	11101110	11101111
-127	-1111111	11111111	10000000	10000001

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Временная диаграмма цифровых сигналов на портах ввода-вывода (фрагмент).
4. Комментированный листинг программы на языке ассемблера.

5. Ответы на контрольные вопросы.

Контрольные вопросы

1. Какими способами можно подключить внешние устройства (светодиод, кнопку) к микроконтроллеру?
2. Как реализуется подсистема прерываний в микроконтроллере AVR?
3. Как программно разрешить или запретить выполнение конкретного прерывания?
4. Какие источники прерываний есть в микроконтроллерах AVR?
5. Как настраиваются внешние прерывания?

Варианты заданий

В описании вариантов символ (*) обозначает, что число ***-y*** вычисляется и представляется в прямом, обратном и дополнительном коде в зависимости от номера студента в составе бригады:

Номер варианта	Правило вычисления ***-y***
Ха	Прямой код
Хб	Обратный код
Хв	Дополнительный код

Варианты логики работы внешних прерываний INT0 и INT1, светодиодов и кнопок на портах PORTA, PORTB, PORTC и PORTD:

Номер варианта	Задание
1	Регистры PORTA–PORTB выполняют роль новогодней гирлянды. Есть следующие 3 режима работы: – на каждом регистре два состояния – вывод чисел *0xFF* и *0x00*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел *0xAA* и *0x55*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел ***y*** и ***-y****, смена состояний с частотой ***x***. Регистры PD4-PD5 отображают номер режима работы (1-3), регистр PD6 отображают номер состояния в конкретном режиме (0/1). Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке

	основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Смена режима работы должна производиться циклически с помощью кнопок PD2 (прерывание INT0) и PD3 (прерывание INT1), в прямом и обратном порядке соответственно, при этом номер активного состояния должен сохраняться. Изменения должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3. Исходное положение: первое состояние, величина ***x*** = *2* Гц, ***y*** = 0xAA, номер режима извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
2	Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число ***y*** и ***-y****, например, если *y*=53=*0x*35=*0b***0011.0101**, то в варианте с обратным кодом «бегущий огонь» выглядит как *0b***0011.0101.1100.1010**). Для каждого состояния активные светодиоды загорается на ***z*** с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: $n_{i+1} = n_i + x \ (mod\ 16)$, где n_i – номер текущего светодиода, n_{i+1} – номер следующего светодиода, ***x*** – величина шага, лежащая в интервале [-3;+3]. Регистры PD0-PD1 отображают номер элемента ***z*** (1-3) в множестве величин задержки {0,25 с; 0,5 с; 1 с}; PD4-PD6 отображают значение ***x*** в прямом коде. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Кнопки PD2 (прерывание INT0) увеличивает величину шага ***x*** на единицу за одно нажатие до +3, после чего устанавливает её равной -3 (одно нажатие – одно изменение).

	Кнопка PD3 (прерывание INT1) циклически изменяет время задержки ***z*** в множестве величин задержки {0,25 с; 0,5 с; 1 с}. Изменение величин шага и задержки должно отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение: величина шага ***x*** = *1*, величина задержки ***z*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно, PORTA = ***y*** = *0x83*.
3	Регистры PORTA–PORTB выполняют роль новогодней гирлянды. Есть следующие 2 режима работы: – на каждом регистре два состояния – вывод чисел *0xFF* и *0x00*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел ***y*** и ***-y****, смена состояний с частотой ***x***. Регистр PD4 отображают номер режима работы (0/1), регистр PD5 отображают номер состояния в конкретном режиме (0/1), регистры PD0-PD1 отображают номер элемента ***x*** в множестве частот. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Смена режима работы должна производиться циклически с помощью кнопки PD2 (прерывание INT0), при этом номер активного состояния должен сохраняться. С помощью кнопки PD3 (прерывание INT1) циклически изменяется величина ***x*** в следующем множестве {0,25 Гц; 0,5 Гц; 1 Гц}. Изменения должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3. Исходное положение первый режим и первое состояние (0xFF на регистрах PORTA-PORTC), величина ***x*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно, ***y*** = 0x55.

4	Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число ***y*** и ***-y****, например, если *y*=53=*0x*35=*0b***0011.0101**, то в варианте с обратным кодом «бегущий огонь» выглядит как *0b***0011.0101.1100.1010**). Для каждого состояния активные светодиоды загорается на 0,2 с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: $n_{i+1} = n_i + x \ (mod\ 16)$, где n_i – номер текущего светодиода, n_{i+1} – номер следующего светодиода, ***x*** – величина шага, лежащая в интервале [-3;+3]. Разряды PD4-PD6 отображают значение ***x*** в прямом коде. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Кнопка PD2 (прерывание INT0) увеличивает величину шага ***x*** до +3, после чего устанавливает её равной -3 (одно нажатие – одно изменение). Кнопка PD3 (прерывание INT1) меняет взаимное расположение ***y*** и ***-y**** в «бегущем огне». Изменения величины шага должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение: PORTA = ***y*** = *0x83*, величина шага ***x*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
5	Регистры PORTA–PORTB выполняют роль новогодней гирлянды. Есть следующие 3 режима работы: – на каждом регистре два состояния – вывод чисел *0xFF* и *0x00*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел *0xAA* и *0x55*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел ***y*** и ***-y****, смена состояний с частотой ***x***.

	Регистры PD0-PD1 отображают номер режима работы (1-3), регистр PD4 отображают номер состояния в конкретном режиме (0/1). Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Смена режима работы должна производиться циклически с помощью кнопок PD2 (прерывание INT0) и PD3 (прерывание INT1), в прямом и обратном порядке соответственно, при этом номер активного состояния должен сохраняться. Изменения должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3. Исходное положение: первое состояние и величина ***x*** = *2* Гц, ***y*** = 0x55, номер режима извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
6	Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число ***y*** и ***-y****, например, если y=53=*0x*35=*0b***0011.0101**, то в варианте с обратным кодом «бегущий огонь» выглядит как *0b***0011.0101.1100.1010**). Для каждого состояния активные светодиоды загорается на ***z*** с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: $n_{i+1} = n_i + x \ (mod\ 16)$, где n_i – номер текущего светодиода, n_{i+1} – номер следующего светодиода, ***x*** – величина шага, лежащая в интервале [-3;+3]. Регистры PD0-PD1 отображают номер элемента ***z*** в множестве величин задержки; PD4-PD6 отображают значение ***x*** в прямом коде. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7,

	после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Кнопки PD2 (прерывание INT0) увеличивает величину шага ***x*** до +3, после чего устанавливает её равной -3 (одно нажатие – одно изменение). Кнопка PD3 (прерывание INT1) циклически изменяет время задержки ***y*** в следующем множестве {0,25 с; 0,5 с; 1 с}. Изменение величин шага и задержки должно отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение: PORTA = ***y*** = *0x73*, величина шага ***x*** = *1*, величина задержки ***z*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
7	Регистры PORTA–PORTB выполняют роль новогодней гирлянды. Есть следующие 3 режима работы: – на каждом регистре два состояния – вывод чисел *0xFF* и *0x00*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел *0xAA* и *0x55*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел ***y*** и ***-y****, смена состояний с частотой ***x***. Регистры PD4-PD5 отображают номер режима работы (1-3), регистр PD6 отображают номер состояния в конкретном режиме (0/1), регистры PD0-PD1 отображают номер элемента ***x*** в множестве частот. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Смена режима работы должна производиться циклически с помощью кнопки PD2 (прерывание INT0), при этом номер активного состояния должен сохраняться.

	С помощью кнопки PD3 (прерывание INT1) циклически изменяется величина ***x*** в следующем множестве {0,25 Гц; 0,5 Гц; 1 Гц}. Изменения должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3. Исходное положение: первый режим и первое состояние (0xFF на регистрах PORTA-PORTC), величина ***x*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно, ***y*** = 0x55.
8	Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число ***y*** и ***-y****, например, если ***y***=53=*0x*35=*0b***0011.0101**, то в варианте с обратным кодом «бегущий огонь» выглядит как *0b***0011.0101.1100.1010**). Для каждого состояния активные светодиоды загорается на 0,2 с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: $n_{i+1} = n_i + x \ (mod\ 16)$, где n_i – номер текущего светодиода, n_{i+1} – номер следующего светодиода, ***x*** – величина шага, лежащая в интервале [-3;+3]. Разряды PD4-PD6 отображают значение ***x*** в прямом коде. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Кнопки PD2 (прерывание INT0) увеличивает величину шага ***x*** до +3, после чего устанавливает её равной -3 (одно нажатие – одно изменение на единицу). Кнопка PD3 (прерывание INT1) меняет взаимное расположение ***y*** и ***-y**** в «бегущем огне». Изменения величины шага должны отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение:

	PORTA = ***y*** = *0x83*, величина шага ***x*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
9	Регистры PORTA–PORTB выполняют роль новогодней гирлянды. Есть следующие 3 режима работы: – на каждом регистре два состояния – вывод чисел *0xFF* и *0x00*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел *0xAA* и *0x55*, смена состояний с частотой ***x***; – на каждом регистре два состояния – вывод чисел ***y*** и ***-y****, смена состояний с частотой ***x***. Регистры PD0-PD1 отображают номер режима работы (1-3), регистр PD4 отображают номер состояния в конкретном режиме (0/1). Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Смена режима работы должна производиться циклически с помощью кнопок PD2 (прерывание INT0) и PD3 (прерывание INT1), в прямом и обратном порядке соответственно, при этом номер активного состояния должен сохраняться. Смена режима должна фиксироваться на PORTD и сохраняться во внутренний EEPROM МК в момент нажатия кнопок. Исходное положение: величина ***x*** = *2* Гц и первое состояние, ***y*** = 0x71, номер режима извлекается из EEPROM, место и формат хранения выбирается самостоятельно.
10	Разрядам регистров PORTA и PORTB ставится в соответствие отрезок [0–15] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7). На данном отрезке отображается «бегущий огонь» из нескольких горящих светодиодов (восьмиразрядное число ***y*** и ***-y****, например, если ***y***=53=*0x*35=*0b***0011.0101**, то в варианте с обратным кодом «бегущий огонь» выглядит как *0b***0011.0101.1100.1010**). Для каждого состояния активные светодиоды загорается на 0,2 с, после чего активным становится другие светодиоды. Номер следующего активного светодиода определяется по формуле: $n_{i+1} = n_i + x \ (mod$

	16), где n_i – номер текущего светодиода, n_{i+1} – номер следующего светодиода, ***x*** – величина шага, лежащая в интервале [-3;+3]. Если величина шага ***x*** > *0*, то PD0 = 0 и PD1 = 1, если величина шага *x* = *0*, то PD0 = 1 и PD1 = 1, если величина шага ***x*** < *0*, то PD0 = 0 и PD1 = 1; PD4-PD5 отображают значение |***x***|. Ввод числа ***y*** должен выполняться на PORTC (одновременным нажатием на несколько кнопок) при нажатой кнопке PD7. Нажатие кнопки PD7 может (а вернее должно) приводить к остановке основного функционала (вывод информации на светодиоды) до момента отпускания всех кнопок на PORTC или отпускания PD7, после чего основной функционал возобновиться с того же режима и состояния, но с новым значением ***y***. Кнопки PD2 (прерывание INT0) и PD3 (прерывание INT1) осуществляют изменение величины шага ***x*** на +1 (mod 4) и -1 (mod 4) соответственно. Изменение величины шага должно отображаться на регистре PORTD и сохраняться во внутренний EEPROM МК в момент нажатия на кнопки PD2 и PD3, изменение логики отображения – со следующего шага «бегущего огня». Исходное положение: PORTA = ***y*** = *0x71*, величина шага ***x*** извлекается из EEPROM, место и формат хранения выбирается самостоятельно.

Лабораторная работа 3

РАБОТА С ТАЙМЕРАМИ-СЧЁТЧИКАМИ

Цель работы

Получение практических навыков по работе с таймерами-счётчиками и применению механизма прерываний.

Теоретические сведения

Микроконтроллеры AVR имеют в своём составе от 1 до 4 таймеров-счётчиков с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника тактовой частоты, и как счётчики внешних событий.

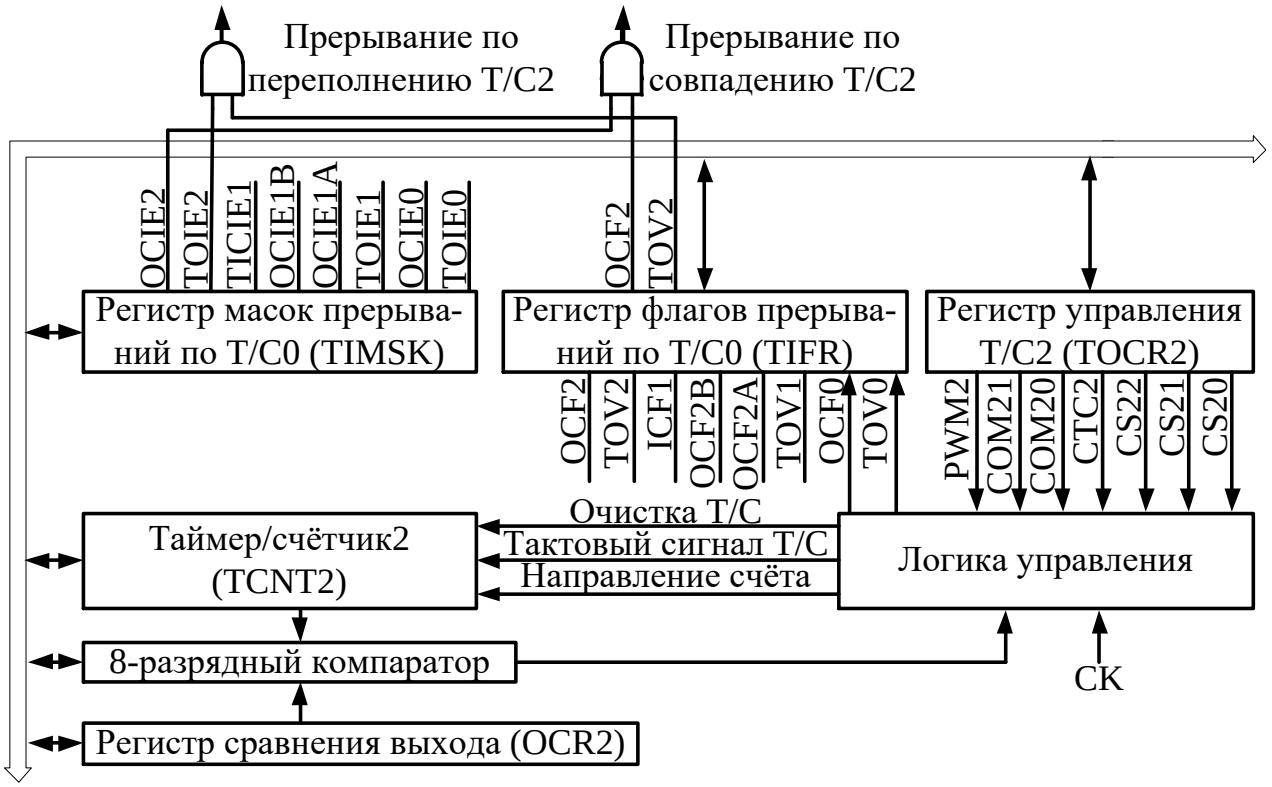


Рис. 1. Схема работы восьмиразрядного таймера TMR0

Таймеры-счётчики можно использовать для точного формирования временных интервалов, подсчёта импульсов на выводах микроконтроллера, формирования последовательности импульсов, тактирования приёмопередатчика последовательного канала связи. В режиме широтно-импульсной модуляции (ШИМ, PWM) таймер-счётчик используется для генерирования сигнала с программируемой частотой и скважностью. Таймеры-счётчики способны вырабатывать запросы

прерываний, переключая процессор на их обслуживание по событиям и освобождая его от необходимости периодического опроса состояния таймеров.

Значение таймера можно прочитать или изменить программным образом в произвольный момент времени. Далее рассматривается конфигурирование таймера T0 микроконтроллера ATmega32.

Таймеры-счётчики

За конфигурацию таймера-счётчика T0 отвечает регистр TCCR0, он определяет источник тактирования таймера, коэффициент предделителя, режим работы таймера-счётчика T0 и поведение вывода OC0.

Биты *CS02*, *CS01*, *CS00* (Clock Select) – определяют источник тактовой частоты для таймера T0 и задают коэффициент предделителя. Все возможные состояния описаны в спецификации контроллера.

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Рис. 2. Регистр TCCR0 (Timer/Counter Control Register)

Биты *WGM01*, *WGM00* (Wave Generator Mode) – определяют режим работы таймера-счётчика T0. Всего их может быть четыре – нормальный режим (Normal), сброс таймера при совпадении (CTC), и два режима широтно-импульсной модуляции (FastPWM и Phase Correct PWM).

WGM01	WGM00	Режим работы таймера-счётчика
0	0	Normal
0	1	PWM, Phase correct
1	0	CTC
1	1	Fast PWM

Биты *COM01*, *COM00* (Compare Match Output Mode) – определяют поведение вывода *OC0*. Если хоть один из этих битов установлен в единицу, то вывод *OC0* перестаёт функционировать как обычный вывод общего назначения и подключается к схеме сравнения таймера счётчика T0. Однако при этом он должен быть ещё настроен как выход.

Поведение вывода *OC0* зависит от режима работы таймера-счётчика T0. В режимах normal и CTC вывод OC0 ведёт себя одинаково, а вот в режимах широтно-импульсной модуляции его поведение отличается.

Бит регистра *TCCR0* – это бит *FOC0* (Force Output Compare). Этот бит предназначен для принудительного изменения состояния вывода *OC0*. Он работает только для режимов Normal и CTC. При установке бита *FOC0* в единицу состояние вывода меняется соответственно значениям битов *COM01*, *COM00*. *FOC0* бит не вызывает прерывания и не сбрасывает таймер в CTC режиме.

Регистр TCNT0 – восьми разрядный счётный регистр. Когда таймер работает, по каждому импульсу тактового сигнала значение TCNT0 изменяется на единицу. В зависимости от режима работы таймера, счётный регистр может или увеличиваться, или уменьшаться.

Регистр TCNT0 можно как читать, так и записывать. Последнее используется, когда требуется задать его начальное значение. Когда таймер работает, изменять его содержимое TCNT0 не рекомендуется, так как это блокирует схему сравнения на один такт.

Регистр OCR0 – восьмиразрядный регистр сравнения. Его значение постоянно сравнивается со счётным регистром TCNT0, и в случае совпадения таймер может выполнять какие-то действия – вызывать прерывание, менять состояние вывода OC0 и т.д. в зависимости от режима работы. Значение OCR0 можно как читать, так и записывать.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Комментированный листинг программы на языке ассемблера.
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Посредством каких регистров производится конфигурирование таймера-счётчика?
2. Какие источники импульсов могут применяться для увеличения таймера-счётчика и для каких целей?
3. В каких режимах могут работать таймеры-счётчики?
4. Как рассчитать начальное значение таймера-счётчика по заданному времени, которое должен отмерить таймер-счётчик до своего переполнения?
5. В чём состоит отличие работы таймера-счётчика в режиме таймера и в режиме счётчика?

Общее описание и требования к программе

Не допускается применение функции delay_ms() и подобных, для реализации временных задержек необходимо использовать прерывания таймеров-счётчиков T0-T2 по переполнению (_OVF) и/или сравнению (_COMP). Точность временных интервалов должна достигаться за счёт правильного выбора предделителей таймеров-счётчиков и применением глобальных переменных. В различных режимах таймеры могут решать различные задачи, например, отсчёт временных промежутков и обеспечение «визуально одновременного» вывода информации на семисегментные индикаторы.

Номер варианта	Задание
1	Секундомер. Программа должна предоставлять возможность запуска секундомера, сохранения зафиксированных значений, отображения текущего и сохранённых значений секундомера на блоке из четырёх семисегментных индикаторов (формат MM.CC, где MM – минуты, CC – секунды). Используемые кнопки: – кнопка PD2 (прерывание INT0) – запуск/останов/возобновление секундомера; – кнопка PD0 – сохранение текущего значения секундомера (работает только при запущенном секундомере); – кнопка PD1 – циклическое пролистывание сохранённых значений на семисегментных индикаторах с первого до последнего (работает только при остановленном секундомере); – кнопка PD3 (прерывание INT1) – останов и сброс секундомера в состояние 0 минут, 0 секунд, удаление сохранённых значений.
2	Модуль настройки ПИН-кода. Программа должна предоставлять возможность по настройке четырёхзначного ПИН-кода. ПИН-код должен храниться в EEPROM, считываться при запуске МК и сохраняться при каждом изменении. Для отображения ПИН-кода используется блок из четырёх семисегментных индикаторов. Ввод четырёхзначного ПИН-кода осуществляется поразрядно, от младшего к старшему. Для изменения значения вводимой цифры ПИН-кода используются

	кнопки PD0 и PD1, позволяющие соответственно увеличить или уменьшить вводимое значение, изменение циклическое, то есть при увеличении цифра 9 переходит в цифру 0 и наоборот, изменение значений происходит следующим образом: – в момент нажатия кнопки значение сразу изменяется на единицу; – если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 каждые 0,25 с. При срабатывании прерывания INT1 (кнопка PD3) во время настройки 1-3 разряда ПИН-кода программа перейдёт к настройке следующего разряда, если выполнялась настройка 4 разряда – программа завершится, сохранив новое значение ПИН-кода в EEPROM. При срабатывании прерывания INT0 (кнопка PD2) во время настройки 2-4 разряда ПИН-кода программа перейдёт к настройке предыдущего разряда. Изначально отображается предыдущее значение ПИН-кода, настраиваемый разряд мигает с частотой 2 Гц.
3	Модуль часов. Программа должна предоставлять возможность настройки текущего времени и отображения времени на блоке из четырёх семисегментных индикаторов (формат «ЧЧ.ММ.» или «ММ.СС», где ЧЧ – часы, ММ – минуты, СС – секунды). Использование внешних прерываний: кнопка PD2 (прерывание INT0) – переключение между режимами настройки и отображения, кнопка PD3 (прерывание INT1) – переход между форматами «ЧЧ.ММ.» или «ММ.СС» в режиме отображения и циклическое переключение между настраиваемыми элементами ЧЧ → ММ → СС → ЧЧ в режиме настройки. В режиме настройки текущий настраиваемый элемент должен мигать с частотой 2 Гц. Изменение значения элемента должно осуществляться с помощью кнопок PD0 и PD1, которые соответственно должны увеличивать и уменьшать значение следующим образом: – в момент нажатия кнопки значение сразу изменяется на единицу; – если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 каждые 0,2 с;

	– если кнопка зажата дольше 4-х секунд, то, начиная со 4-й секунды, значение начинает изменяться на 1 каждые 0,1 с; после отпускания кнопки изменение немедленно прекращается.
4	Модуль настройки ПИН-кода. Программа должна предоставлять возможность по настройке четырёхзначного ПИН-кода. ПИН-код должен храниться в EEPROM, считываться при запуске МК и сохраняться при каждом изменении. Для отображения ПИН-кода используется блок из четырёх семисегментных индикаторов. Для ввода цифры ПИН-кода используются кнопки, подключённые к PORTA и PORTB: PB0 – цифра 0, PB1 – 1, PB2 – 2, PB3 – 3, PB4 – 4, PB5 – 5, PB6 – 6, PB7 – 7, PA6 – 8, PA7 – 9. Ввод четырёхзначного ПИН-кода осуществляется поразрядно, от младшего к старшему, если в течение 10 секунд с момента ввода 1-3 цифры ПИН-кода не была введена следующая цифра программа возвращается к началу ввода ПИН-кода, изображение на семисегментных индикаторах замещается сохранённым ранее в EEPROM. Изначально отображается предыдущее значение ПИН-кода, младший разряд мигает с частотой 2 Гц, сразу после ввода нового значения для младшего разряда (нажатия на соответствующую кнопку PORTA / PORTB) значение разряда обновляется, а мигание переносится на следующий разряд. После ввода старшего разряда новое значение ПИН-кода сохраняется в EEPROM, настройка ПИН-кода завершается. При срабатывании прерывания INT0 (кнопка PD2) во время настройки 2-4 разряда ПИН-кода программа перейдёт к настройке предыдущего разряда. При срабатывании прерывания INT1 (кнопка PD3) во время настройки 1-3 разряда ПИН-кода программа перейдёт к настройке следующего разряда, если выполнялась настройка 4 разряда – программа завершится, сохранив новое значение ПИН-кода в EEPROM.
5	Таймер. Программа должна предоставлять возможность настройки, запуска и приостановки таймера, отображения настраиваемого и текущего значений таймера на блоке из четырёх семисегментных индикаторов (формат «ММ.СС», где ММ – минуты, СС – секунды). Использование внешних

	прерываний: кнопка PD2 (прерывание INT0) – запуск/приостановка/возобновление таймера, кнопка PD3 (прерывание INT1) – переход между режимами настройки и работы таймера. Изменение значения таймера в режиме настройки должно осуществляться с помощью кнопок PD0 и PD1, которые соответственно должны увеличивать и уменьшать значение следующим образом: – в момент нажатия кнопки значение сразу изменяется на одну секунду; – если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 секунду каждые 0,2 с; – если кнопка зажата дольше 4-х секунд, то, начиная со 4-й секунды, значение начинает изменяться на 1 секунду каждые 0,1 с; – после отпускания кнопки изменение немедленно прекращается. Когда таймер доходит до нулевого значения отсчёт прекращается, на семисегментном индикаторе отображается значение 00.00, мигая с частотой 2 Гц.
6	Модуль часов. Программа должна предоставлять возможность настройки текущего времени и отображения времени на блоке из четырёх семисегментных индикаторов (формат «ЧЧ.ММ.» или «ММ.СС», где ЧЧ – часы, ММ – минуты, СС – секунды). Использование внешних прерываний: кнопка PD2 (прерывание INT0) – переключение между режимами настройки и отображения, кнопка PD3 (прерывание INT1) – переход между форматами «ЧЧ.ММ.» или «ММ.СС» в режиме отображения и циклическое переключение между настраиваемыми элементами ЧЧ → ММ → СС → ЧЧ в режиме настройки. В режиме настройки текущий настраиваемый элемент должен мигать с частотой 2 Гц. Изменение значения элемента должно осуществляться с помощью кнопок PD0 и PD1, которые соответственно должны увеличивать и уменьшать значение следующим образом: – в момент нажатия кнопки значение сразу изменяется на единицу;

	– если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 каждые 0,2 с; – если кнопка зажата дольше 4-х секунд, то, начиная со 4-й секунды, значение начинает изменяться на 1 каждые 0,1 с; после отпускания кнопки изменение немедленно прекращается.
7	Модуль проверки ПИН-кода. Программа должна предоставлять возможность по проверке введённого ПИН-кода на соответствие заданному. ПИН-код должен храниться в EEPROM и считываться при запуске МК (запись ПИН-кода в EEPROM следует выполнять с помощью AVRFlash). Для отображения проверяемого ПИН-кода используется блок из четырёх семисегментных индикаторов. Для ввода цифры ПИН-кода используются кнопки, подключённые к PORTA и PORTB: PB0 – цифра 0, PB1 – 1, PB2 – 2, PB3 – 3, PB4 – 4, PB5 – 5, PB6 – 6, PB7 – 7, PA4 – 8, PA5 – 9. При вводе неправильного ПИН-кода загорается светодиод PA6 на 20 секунд, после чего программа возвращается к началу ввода ПИН-кода для проверки. При вводе неправильного ПИН-кода три раза подряд загораются светодиод PA7 и PA6, а программа перестаёт работать. При вводе правильного ПИН-кода загорается светодиод PA7, вернуться к началу режима проверки ПИН-кода (завершить текущий сеанс) можно с помощью прерывания INT1 (кнопка PD3) – выполнится очистка семисегментных индикаторов и погаснет светодиод PA7. Ввод четырёхзначного ПИН-кода осуществляется поразрядно, если в течение 7 секунд с момента ввода 1-3 цифры ПИН-кода не была введена следующая цифра программа возвращается к началу ввода ПИН-кода для проверки. Ввод ПИН-кода осуществляется от младшего разряда к старшему. В начале работы программы на семисегментном индикаторе нет цифр, сразу после ввода каждой цифры она дописывается на семисегментный индикатор.
8	Секундомер. Программа должна предоставлять возможность запуска секундомера, сохранения зафиксированных значений, отображения текущего и сохранённых значений секундомера на блоке из четырёх

	семисегментных индикаторов (формат ММ.СС, где ММ – минуты, СС – секунды). Используемые кнопки: – кнопка PD2 (прерывание INT0) – запуск/останов/возобновление секундомера; – кнопка PD0 – сохранение текущего значения секундомера (работает только при запущенном секундомере); – кнопка PD1 – циклическое пролистывание сохранённых значений на семисегментных индикаторах с первого до последнего (работает только при остановленном секундомере); – кнопка PD3 (прерывание INT1) – останов и сброс секундомера в состояние 0 минут, 0 секунд, удаление сохранённых значений.
9	Модуль проверки ПИН-кода. Программа должна предоставлять возможность по проверке введённого ПИН-кода на соответствие заданному. ПИН-код должен храниться в EEPROM и считываться при запуске МК (запись ПИН-кода в EEPROM следует выполнять с помощью AVRFlash). Для отображения проверяемого ПИН-кода используется блок из четырёх семисегментных индикаторов. настраиваемый разряд мигает с частотой 4 ГЦ. Ввод четырёхзначного ПИН-кода осуществляется поразрядно, от младшего к старшему. Для изменения значения вводимой цифры ПИН-кода используются кнопки PD0 и PD1, позволяющие соответственно увеличить или уменьшить вводимое значение, изменение циклическое, то есть при увеличении цифра 9 переходит в цифру 0 и наоборот, изменение значений происходит следующим образом: – в момент нажатия кнопки значение сразу изменяется на единицу; – если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 каждые 0,25 с. При срабатывании прерывания INT0 (кнопка PD2) во время настройки 2-4 разряда ПИН-кода программа перейдёт к настройке предыдущего разряда. При срабатывании прерывания INT1 (кнопка PD3) во время ввода 1-3 разряда ПИН-кода программа перейдёт к вводу следующего разряда, если выполнялась настройка 4 разряда – ввод ПИН-кода завершится,

	а программа сверит введённое значение ПИН-кода со значением в EEPROM. При вводе неправильного ПИН-кода загорается светодиод PA6 на 20 секунд, после чего программа возвращается к началу ввода ПИН-кода для проверки. При вводе неправильного ПИН-кода три раза подряд загораются светодиод PA7 и PA6, а программа перестаёт работать. При вводе правильного ПИН-кода загорается светодиод PA7, программа завершает работу (переходит в бесконечный цикл, отключаются прерывания INT0 и INT1. В начале работы программы на семисегментном индикаторе отображено значение «0000».
10	Таймер. Программа должна предоставлять возможность настройки, запуска и приостановки таймера, отображения настраиваемого и текущего значений таймера на блоке из четырёх семисегментных индикаторов (формат «MM.CC», где MM – минуты, CC – секунды). Использование внешних прерываний: кнопка PD2 (прерывание INT0) – запуск/приостановка/возобновление таймера, кнопка PD3 (прерывание INT1) – переход между режимами настройки и работы таймера. Изменение значения таймера в режиме настройки должно осуществляться с помощью кнопок PD0 и PD1, которые соответственно должны увеличивать и уменьшать значение следующим образом: – в момент нажатия кнопки значение сразу изменяется на одну секунду; – если кнопка зажата дольше 2-х секунд, то, начиная со 2-й секунды, значение начинает изменяться на 1 секунду каждые 0,2 с; – если кнопка зажата дольше 4-х секунд, то, начиная со 4-й секунды, значение начинает изменяться на 1 секунду каждые 0,1 с; – после отпускания кнопки изменение немедленно прекращается. Когда таймер доходит до нулевого значения отсчёт прекращается, на семисегментном индикаторе отображается значение 00.00, мигая с частотой 2 Гц.

Лабораторная работа 4

ОРГАНИЗАЦИЯ АНАЛОГОВОГО ВВОДА-ВЫВОДА: АЦП И ШИМ

Цель работы

Получение практических навыков по работе с аналого-цифровым преобразователем и таймерами-счётчиками. Изучение принципов применения широтно-импульсной модуляции в современных цифровых устройствах.

Теоретические сведения

Аналого-цифровой преобразователь

ATMega32 содержит в себе 10-битовый аналого-цифровой преобразователь (АЦП, ADC), вход которого может быть соединён с одним из восьми выводов PORTA. АЦП ATmega32, как и любому другому АЦП, нужно опорное напряжение для сравнения с входным напряжением (если измеряемое равно опорному, то получаем максимальный код в двоичном виде). Опорное напряжение подаётся на вывод *ADRef* или может использоваться внутренний генератор с фиксированным напряжением 2,65 В. Полученный результат можно представить в таком виде: *ADC = Vin * 1024/Vref*.

АЦП включается установкой бита *ADEN* в регистре *ADCSRA*. После преобразования, 10-битный результат оказывается в 8-битных регистрах *ADCL* и *ADCH*. По умолчанию, младший бит результата находится справа (так называемое правое ориентирование). Возможно изменение порядка следования битов на лево-ориентированное путём установки бита *ADLAR* в регистре *ADMUX*. В таком случае для получения 8-битного результата достаточно прочитать регистр *ADCH*. В противном случае, требуется сначала прочитать регистр *ADCL*, а потом *ADCH*.

Одиночное преобразование может быть вызвано записью бита *ADSC* в регистр *ADCSRA*. Этот бит остаётся установленным всё время, занимаемое преобразованием. Когда преобразование закончено, бит автоматически устанавливается в ноль. Можно также начинать преобразования по событиям из разных источников. Модуль АЦП также может работать в режиме «свободного полёта». В таком случае АЦП

постоянно производит преобразование и обновляет регистры *ADCH* и *ADCL* новыми значениями.

Для выполнения преобразования модулю АЦП необходима тактовая частота. Чем выше эта частота, тем быстрее будет происходить преобразование (оно, обычно, занимает 13 тактов, первое преобразование занимает 25 тактов). Но чем выше частота (и выше скорость преобразования), тем менее точным получается результат. Для получения максимально точного результата, модуль АЦП должен тактироваться частотой в пределах от 50 до 200 КГц. Если необходим результат с точностью менее 10 бит, то можно использовать частоту больше 200 КГц. Модуль АЦП содержит делитель частоты, чтобы получать нужную тактовую частоту для преобразования из частоты процессора.

7	6	5	4	3	2	1	0
REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Рис. 11 Регистр ADMUX

Регистр *ADMUX* задаёт входной контакт PORTA для подключения АЦП, ориентирование результата и выбор опорной частоты. Если установлен бит *ADLAR*, то результат лево-ориентированный. Опорная частота от внутреннего генератора задаётся выставленными в единицу битами *REFS1* и *REFS0*. Если оба бита сброшены, то опорная частота берётся от контакта *AREF*. Если *REFS1*=0 и *REFS0*=1, то опорная частота берётся от *AVCC* с внешним конденсатором, подключённым к *AREF*.

Номер канала АЦП *ADC0, ADC1, ..., ADC7*, с которого будет получаться сигнал, соответствует значениям битов *MUX4...1* 0x00000, 0x00001, …, 0x00111.

7	6	5	4	3	2	1	0
ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Рис. 22 Регистр ADCSRA

АЦП включается установкой бита *ADEN*.

Запись единицы в *ADSC* регистра ADCSRA запускает цикл преобразования.

ADIF – флаг прерывания АЦП. Этот бит устанавливается в единицу, когда АЦП завершено преобразование и в регистрах *ADCL* и *ADCH*

находятся актуальные данные. Этот флаг устанавливается даже в том случае, если прерывания запрещены. Это необходимо для случая программного опроса АЦП. Если используются прерывания, то флаг сбрасывается автоматически. Если используется программный опрос, то флаг может быть сброшен записью логической единицы в этот бит.

Если в бите *ADIE* установлена единица, и прерывания разрешены глобально, то при окончании преобразования будет выполнен переход по вектору прерывания от АЦП.

Широтно-импульсная модуляция

Широтно-импульсная модуляция, или ШИМ, это операция получения изменяющегося аналогового значения посредством цифровых устройств. Устройства используются для получения прямоугольных импульсов – сигнала, который постоянно переключается между максимальным и минимальным значениями. Данный сигнал моделирует напряжение между максимальным значением (5 В) и минимальным (0 В), изменяя при этом длительность времени включения 5 В относительно включения 0 В. Длительность включения максимального значения называется шириной импульса. Для получения различных аналоговых величин изменяется ширина импульса.

Рисунок 3 показывает примеры различных сигналов ШИМ. Рисунок а) показывает выход ШИМ с 90% скважностью. Таким образом, сигнал имеет высокий уровень в течение 90% от периода и низкий – в течение остальных 10%. Рисунки в) показывают выходы ШИМ с 50% скважностью. Эти два выхода ШИМ кодируют различные значения аналогового сигнала, в 90% и 50% от полной силы. Если, например, напряжение Vcc будет равно 5 В, а скважность 10%, то результатом будет 0.5 В аналогового сигнала.

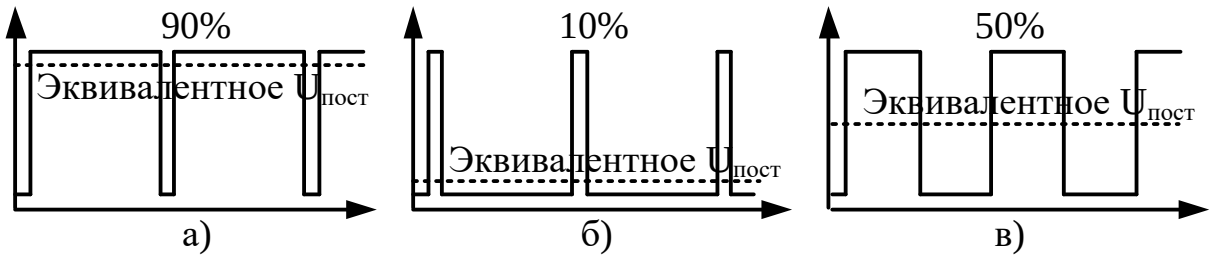


Рис. 33. Примеры широтно-импульсной модуляции

Широтно-импульсная модуляция реализуется с помощью настроек таймеров-счётчиков.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Комментированный листинг программы на языке ассемблера.
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Посредством каких регистров производится настройка АЦП?
2. В каких режимах может работать АЦП?
3. Какие порты и разряды портов микроконтроллера ATmega32 могут обрабатывать входящие аналоговые сигналы?
4. Какими способами реализуется ШИМ?
5. Как настроит ШИМ с помощью таймера-счётчика?

Общее описание и требования к программе

Программа должна осуществлять два режимы работы:

– демонстрация гирлянды в соответствии с вариантом задания;

– настройка параметров работы гирлянды.

Вывод PA5 должен быть настроен на ввод (DDRA=0xdf), для работы гирлянды он игнорируется (изменения в логику работы гирлянды не вносятся). Считывание значения аналогового сигнала должно производиться **с помощью прерывания АЦП (ADC)**.

Переключение между режимами должно осуществляться циклически с помощью кнопки PD2 (прерывание INT0). Переключение между настраиваемыми параметрами должно осуществляться циклически с помощью кнопки PD3 (прерывание INT1). Имя изменяемого параметра должно отображаться на первых (одном, двух или трёх, в зависимости от параметра) семисегментных индикаторах, на последующих индикаторах должно отображаться значение соответствующего параметра в шестнадцатеричной системе счисления. На последнем индикаторе имени параметра в качестве разделителя должна гореть точка. Имя параметра должно отображаться постоянно, а значение циклически загораться и гаснуть с периодами этих состояний 0,5 с. Изменение значений должно осуществляться с помощью потенциометра, подключённого к выводу PA5, следующим образом:

– крайнее левое положение соответствует нижней границе допустимого диапазона, крайнее правое – верхней;

– при повороте потенциометра значение на семисегментном индикаторе изменяется незамедлительно.

Яркость светодиодов PD7 и PD4 при помощи значения скважности (широтно-импульсная модуляция, режим fast-PWM таймеров T1 и T2) должна показываться степень отдалённости настраиваемого параметра от крайних значений. Например, если настраиваемый параметр принимает значения [1-5], а текущая настройка равна двум, то соответственно скважность OC2 равна TOP * 1 / 4, а OC1B равна TOP * 3 / 4. В том же примере при настройке параметра в 1 значение OC2 будет равно величине TOP, а OC1B равно нулю.

Логики работы гирлянды

1. Разрядам регистров PORTA, PORTB и PORTC ставится в соответствие отрезок [0–23] (0-7 ~ PA0–PA7, 8-15 ~ PB0-PB7, 16-23 ~ PC0-PC7). На данном отрезке отображается «бегущий огонь» из набора горящих светодиодов. Для гирлянды существует базовое значение (набор одновременно горящих светодиодов), которое с определённой частотой циклически сдвигается на определённый шаг. Параметрами гирлянды являются:

– базовое значение (*b*) – положительное трёхбайтное число, которое циклически сдвигается и отображается на регистрах PORTA–PORTC. Параметрами являются числа *b0*, *b1* и *b2* (отдельные байты базового значения);

– величина шага (*h*), на которую циклически сдвигается базовое значение каждый такт гирлянды;

– частота смены состояний (*p*);

– начальное смещение (*d*), определяющее с каким значением смещения начинает работать гирлянда при запуске программы.

Гирлянда работает по следующему правилу: $\{s_i = (s_{i-1} << h);\ s_0 = (b << d)\}$. Примеры отображения настроек на семисегментном индикаторе приведены на рисунке 1.

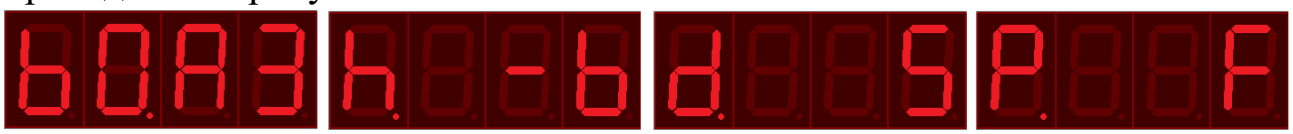


Рисунок 1. Примеры настройки параметров гирлянды №1

2. На каждом из регистров PORTA, PORTB и PORTC отображается «бегущий огонь» из набора горящих светодиодов. Для каждого порта существует базовое значение (набор одновременно горящих светодиодов), которое с определённой частотой (для каждого регистра своей) циклически сдвигается на определённый шаг (для каждого регистра свой). Параметрами гирлянды являются:

– базовое значение (*b*) – положительное число, которое циклически сдвигается и отображается на соответствующем регистре;

– величина шага (*h*), на которую циклически сдвигается базовое значение каждый такт гирлянды;

– частота смены состояния (*p*);

– начальное смещение (*d*), определяющее с каким значением смещения начинает работать гирлянда при запуске программы.

Все параметры задаются для регистров PORTA, PORTB и PORTC в виде «ba.», «bb.», «bc.» и т.д. Примеры отображения настроек на семисегментном индикаторе приведены на рисунке 2.

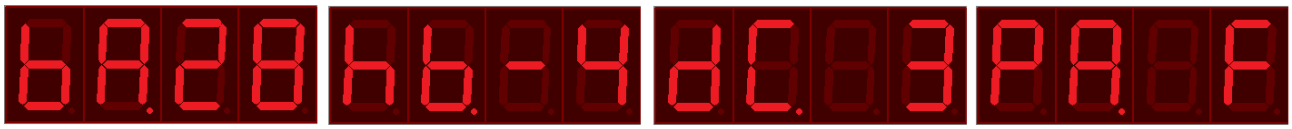


Рисунок 2. Примеры настройки параметров гирлянды №2

3. На каждом из регистров PORTA, PORTB и PORTC циклически отображаются комбинаций горящих светодиодов из фиксированного массива значений. Существует общей для всех регистров массив из пяти значений (наборов одновременно горящих светодиодов). Смена состояния гирлянды на каждом порту заключается в циклической последовательной смене индекса отображаемого элемента массива на определённую величину шага. Параметрами гирлянды являются:

– массив значений {*b0*; *b1*; …; *b4*} – положительных чисел, которые могут отображаться на регистрах;

– величина шага (*h*), на которую циклически увеличивается индекс отображаемого элемента массива *b* каждый такт гирлянды, для каждого регистра значение шага своё;

– массив временных интервалов (задаётся обратная величина – частота) смены состояния (*p0*; *p1*; …; *p4*);

– начальный индекс (*d*), определяющее с какого элемента массива b начинает работать гирлянда на соответствующем регистре.

Параметры отличные для регистров PORTA, PORTB и PORTC задаются в виде «hb.», «dc.» и т.д. Примеры отображения настроек на семисегментном индикаторе приведены на рисунке 3.

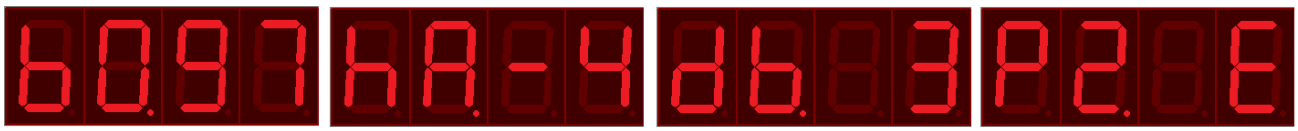


Рисунок 3. Примеры настройки параметров гирлянды №3

Варианты заданий

Вариант	Логика гирлянды	Настраиваемые параметры		Значения неизменяемых параметров
1	1	а	h є [-4; 4]	b1=0; b2=0; d=0
		б	p є [1; 7] раз в 2 секунды	**p=3**
		в	b0 є [0; 127]	**b0=1**
2	2	а	ha є [-3; 3]	bb=1; hb=1; pb=3; db=0;
		б	pa є [1; 6] раз в 2 секунды	bc=7; hc=2; pc=5; dc=1;
		в	ba є [0; 100]	da=0; **pa=1; ba=3**
3	3	а	b0 є [0; 255]	b1=1; b2=2; b3=3; b4=4;
		б	p0 є [1; 10] раз в 2 секунды	hb=1; hc=2; p1=1; p2=2; p3=3;
		в	ha є [-2; 2]	p4=4; da=db=dc=0; **p0=f; ha=3**
4	1	а	p є [1; 5] раз в 2 секунды	b1=0; b2=0; h=1
		б	b0 є [0; 255]	**b0=1**
		в	d є [0; 10]	**d=0**
5	2	а	pb є [1; 8] раз в 2 секунды	ba=3; ha=2; pa=4; da=0;
		б	db є [0; 3]	bc=6; hc=1; pc=6; dc=1;
		в	bb є [0; 127]	hb=1; **db=0; bb=f**
6	3	а	p1 є [1; 3] раз в 2 секунды	b0=1; b3=3; b4=4; ha=3; hb=1;
		б	b1 є [128; 255]	hc=2; p0=1; p2=2; p3=3; p4=4;
		в	b2 є [0; 64]	da=db=dc=0; **b1=aa; b2=7**
7	1	а	d є [0; 7]	b1=0; b2=0; h=1
		б	p є [1; 9] раз в 2 секунды	**p = 3**
		в	b0 є [0; 63]	**b0=1**
8	2	а	ha є [-4; 4]	ba=1; ha=1; pa=2; da=0;
		б	bc є [0; 255]	bb=5; hb=2; pb=3; db=1;
		в	dc є [0; 2]	pc=4; **bc=3; dc=0**
9	3	а	ha є [-2; 2]	b1=1; b2=2; b3=3; b4=4; hb=1;
		б	da є [0; 3]	hc=2; p0=1; p1=2; p2=3; p3=4;
		в	b0 є [0; 255]	p4=5; db=dc=0; **da=1; b0=7**
10	1	а	h є [-3; 3]	b1=0; b2=0; p=5
		б	b0 є [0; 255]	**b0=1**
		в	d є [0; 3]	**d=0**

Лабораторная работа 6

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА И ПЭВМ. ПРОТОТИП ЦИФРОВОГО ОСЦИЛЛОГРАФА

Цель работы

Получение практических навыков по организации взаимодействия микроконтроллера с компьютером. Совершенствование навыков по организации аналогового ввода и работе с UART-интерфейсом.

Теоретические сведения

Взаимодействие с COM-портом со стороны ПК

Таблица 1 — Основные функции работы с COM-портом в ОС MS Windows

Структура для работы с COM-портом: DCB PORT_dcb;
Открытие COM-порта: ComPort = CreateFile("COM3", GENERIC_READ | GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
Считывание настроек COM-порта: ComFlag = GetCommState(ComPort, &PORT_dcb);
Очистка входных и выходных буферов: PurgeComm(ComPort, PURGE_TXABORT | PURGE_RXABORT | PURGE_TXCLEAR | PURGE_RXCLEAR);
Настройка COM-порта: PORT_dcb.BaudRate = Speed; PORT_dcb.StopBits = ONESTOPBIT; PORT_dcb.fParity = NOPARITY; PORT_dcb.ByteSize = 8;
Установка задержки COM-порта: TimeOuts.ReadTotalTimeoutConstant = 1000; SetCommTimeouts(ComPort, &TimeOuts);
Запись настроек COM-порта: ComFlag = SetCommState(ComPort, &PORT_dcb);
Очистка ошибок: ClearCommError(ComPort, &Error, &Stat);
Приём данных с COM-порта: ReadFile(ComPort, receive_buffer, 4, &comb, NULL);
Отправка данных через COM-порт: WriteFile(ComPort, send_buffer, 5, &comb, NULL);

Порядок выполнения работы

1. Разработать программный прототип осциллографа, переводящего уровень напряжения (аналогового сигнала) на входе АЦП микроконтроллера в точку на графике в приложении для ПК с учётом дополнительных требований в соответствии с вариантом задания.

Программа для микроконтроллера должна осуществлять периодическое считывание 10-разрядного значения напряжения U на входе АЦП и отсылку его в последовательный порт. Кроме того, программа должна иметь возможность принимать из последовательного порта значение частоты дискретизации / периода считывания аналогового напряжения (t_{ADC}) и изменять величину частоты / периода отправки данных на ПК (t_{UART}).

Приложение для ПК должно позволять запускать и останавливать осциллограф, осуществлять получение числа из COM-порта и отображать график уровня напряжения на входе АЦП. Приложение должно иметь интерфейсы для задания частоты дискретизации (t_{ADC}), частоты отправки данных с МК на ПК (t_{UART}), выбора используемого COM-порта, скорости его работы* и формата кадра* (число информационных битов и контроль чётности (odd, even, none)). Параметры, отмеченные *, должны работать только в соответствующих вариантах. Приложение должно позволять пользователю изменять перечисленные настройки во время работы осциллографа (желательно использование нескольких потоков).

Протокол взаимодействия ПК и МК должен предусматривать обмен следующими типами сообщений (символом «*» отмечены типы сообщений, необходимые в отдельных вариантах):

№	Тип сообщения	Направление
1	Запуск осциллографа	ПК → МК
2	Остановка осциллографа	ПК → МК
3	Передача фрагмента 10-разрядного значения напряжения (в каждом передаваемом фрагменте должно быть указание на место фрагмента в 10-разрядном значении)	ПК ← МК
4	Передача нового значения частоты отправки данных на ПК t_{UART}	ПК → МК
5*	Запрос сохранённого значения частоты отправки данных на ПК t_{UART}	ПК → МК
6*	Передача сохранённого значения частоты отправки данных на ПК t_{UART}	ПК ← МК
7	Передача нового значения частоты дискретизации t_{ADC}	ПК → МК
8*	Запрос сохранённого значения частоты дискретизации t_{ADC}	ПК → МК

9*	Передача сохранённого значения частоты дискретизации t_{ADC}	ПК ← МК
10*	Запрос статистики (количество измерений, максимальное и минимальное значение напряжения)	ПК → МК
11*	Передача статистики (количество измерений, максимальное и минимальное значение напряжения)	ПК ← МК
12*	Инициализация сохранения статистики (количество измерений напряжения/ максимальное значение напряжения/минимальное значение напряжения) в EEPROM	ПК → МК
13*	Очистка статистики (в ОЗУ и в EEPROM)	ПК → МК
14*	Инициализация изменения настроек интерфейса (скорость UART и формат кадра – количество информационных битов и контроль чётности)	ПК → МК
15*	Инициализация изменения настроек интерфейса (скорость UART и формат кадра – количество информационных битов и контроль чётности)	ПК ← МК
16*	Инициализация сохранения параметров (t_{ADC}, t_{UART}, настройки интерфейса) в EEPROM	ПК → МК
17*	Передача размера блока (длина массива значений аналогового напряжения)	ПК → МК
18*	Передача размера блока (длина массива значений аналогового напряжения)	ПК ← МК

Формат протокола разрабатывается самостоятельно. Наиболее простой способ организации протокола подразумевает выделение нескольких битов в пакете для указания типа сообщения. При передаче сообщений в несколько байтов рекомендуется предусмотреть дополнительные пакеты между сообщениями – либо пустые пакеты, либо пакеты с контрольными суммами, либо пакеты-подтверждения, передаваемые в обратную сторону.

2. Скомпилировать и записать программу в контроллер.

3. Проверить правильность работы программного комплекса.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).

2. Блок-схемы алгоритмов программ для микроконтроллера и ПК.

3. Комментированные листинги программы для микроконтроллера на языках Си и ассемблер и листинг программы для ПК на языке Си.

4. Список всех встречающихся в программе состояний регистров конфигурации АЦП и UART с пояснением всех разрядов регистров.

5. Выводы по работе.

Варианты заданий

Номер варианта	Задание
1	МК должен осуществлять подсчёт статистики (количество измерений напряжения, максимальное и минимальное значение напряжения). На семисегментный индикатор должна постоянно выводиться текущая статистика (количество измерений напряжения/ максимальное значение напряжения/минимальное значение напряжения), переключение между выводимыми значениями осуществляется с помощью прерывания int0. По прерыванию int1 текущие значения статистики сохраняются в EEPROM. При включении МК исходные значения считываются из EEPROM.
2	На семисегментном индикаторе должны выводиться считанное с АЦП значение, величина t_{ADC} (или обратная к ней) и величина t_{UART} (или обратная к ней) в виде бегущей строки, параметры должны разделяться точкой. Пример вывода: «762.58.239» (красным выделен фрагмент на индикаторе, в следующий момент «762.58.239»). По прерыванию int0 выводимое сообщение должно возвращаться к первому символу («762.58.239»).
3	По прерыванию int0 должна циклически изменяться скорость работы USART: 9600, 38400 и 57600 бод (одно нажатие – одно изменение). Перед изменением скорости МК отправляет соответствующее сообщение на ПК, после чего оба устройства изменяют настройки интерфейсов.
4	Передача значения напряжения должна производиться не отдельными значениями, а массивами настраиваемой длины. Программа на МК заполняет массив значений и только после этого отправляет значения на ПК. В приложении для ПК должен быть интерфейс для ввода длины массива, кнопка для отправки данного параметра на МК. При запуске МК считывает значение длины массива из EEPROM. При запуске осциллографа (получении соответствующего сообщения от ПК) в ответ МК посылает длину массива. При получении сообщения об изменении длины массива МК сохраняет полученное значение в EEPROM.
5	МК должен осуществлять подсчёт статистики (количество измерений напряжения, максимальное и минимальное значение напряжения). Приложение должно иметь интерфейс для вывода статистики, кнопку для запроса текущей статистики от МК, кнопку для записи статистики в EEPROM и кнопку для обнуления статистики в ОЗУ и EEPROM МК. При включении МК значения статистики считываются из EEPROM.
6	Считывание данных с UART-интерфейса должно осуществляться с помощью прерывания USART_RXC. Необходимо реализовать обработку двухбайтных величин t_{ADC} и t_{UART}. Прерывание int0 сбрасывает значение

	величин t_{ADC} и t_{UART} в значение по умолчание (выбирается самостоятельно) и передаёт соответствующие сообщения на ПК.
7	На семисегментном индикаторе должно постоянно выводиться текущее значение t_{ADC} и t_{UART}, переключение между выводимыми значениями осуществляется с помощью прерывания int0 (одно нажатие – одно изменение). По прерыванию int1 текущее отображаемое значение увеличивается на единицу (одно нажатие – одно изменение), новое значение передаётся на ПК.
8	На семисегментном индикаторе должно постоянно выводиться текущее значение напряжения на входе АЦП. Для организации периодического считывания значения напряжения (величина t_{ADC}) необходимо использовать прерывание timer1_cmp. Считывание данных с UART-интерфейса должно осуществляться с помощью прерывания USART_RXC.
9	При изменении параметров UART-интерфейса (скорость, размер кадра и контроль чётности) в приложении на ПК происходит передача сообщения с ПК на МК с указанием новых параметров до изменения самих параметров, после чего оба устройства изменяют свои настройки интерфейсов. По прерыванию int0 МК информирует ПК об установке значений по умолчанию (скорость 9600 бод, 8 информационных битов и выключенный контроль чётности), после чего оба устройства изменяют настройки интерфейсов.
10	При включении МК из памяти EEPROM считываются сохранённые значения величин t_{ADC} и t_{UART}. При запуске осциллографа (передача соответствующего сообщения с ПК на МК) МК должен в ответ прислать свои настройки величин t_{ADC} и t_{UART}. Приложение должно иметь кнопку для записи текущих значений указанных величин в EEPROM.

Лабораторная работа 5

РАБОТА С ВНЕШНИМИ ИНТЕРФЕЙСАМИ: UART И I2C

Цель работы

Изучение структуры данных, пересылаемых через последовательный порт. Получение практических навыков программирования встроенного в микроконтроллер UART модуля. Получение практических навыков по работе с интерфейсом I^2C и принципами работы памяти EEPROM.

Теоретические сведения

Внешние интерфейсы, последовательная передача данных

Устройства на базе микроконтроллеров часто используются в качестве периферийных устройств сбора и первичной обработки информации, работающих в составе более сложной информационной системы, включающей в себя большую ЭВМ, предназначенную для решения более сложных задач анализа и хранения собранной информации.

Существует несколько способов подключения устройств к ЭВМ, в зависимости от требуемой скорости передачи. Подключение к IBM PC устройств с относительно низким быстродействием осуществляется через последовательный порт передачи данных (COM-порт).

Главное различие последовательной и параллельной передачи данных заключается в том, что при последовательной передаче информации в единичный интервал времени передаётся только один бит. Для параллельной передачи данных требуется наличие проводников в количестве, равном количеству одновременно передаваемых бит.

Достоинство последовательной передачи данных заключается в малом количестве проводов, необходимых для организации передачи, а недостаток – это относительно низкая скорость передачи информации.

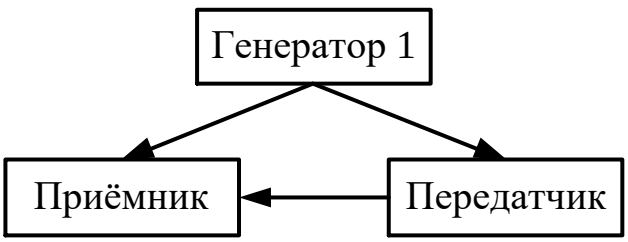


Рис. 1. Схема соединения при синхронной передаче

Существует две разновидности последовательного способа передачи данных – синхронный и асинхронный.

Необходимое условие синхронной передачи данных – наличие одного синхрогенератора для приёмника и передатчика.

Достоинства: высокая скорость передачи.

Недостаток: необходимость специального синхропровода для передачи синхросигнала (для синхронной работы приёмника и передатчика).

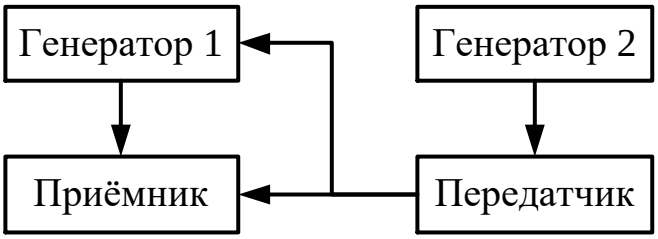


Рис. 2. Схема соединения при асинхронной передаче

Генератор приёмника периодически подстраивается под генератор передатчика по специальным сигналам синхронизации. Для подстройки используют специальную структуру передаваемых сигналов. Обычно синхросигнал предшествует началу передачи блока данных и называется стартовым битом.

Данные в последовательный порт передаются пакетами. Структура пакета данных изображена на рисунке.

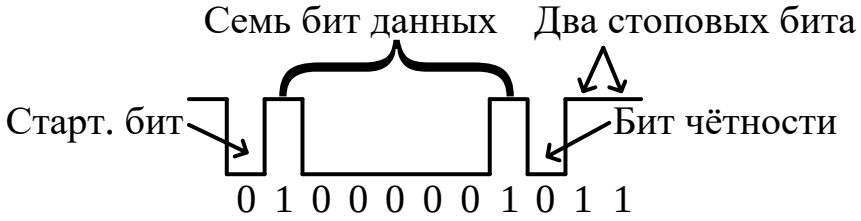


Рис. 3. Структура пакета

Из рисунка видно, что исходное состояние линии последовательной передачи данных – это уровень логической единицы. Стартовый бит служит для формирования перехода из единицы в ноль, который означает начало передачи данных. Далее передаются биты данных (от 5 до 8 в зависимости от выбранного формата данных), далее возможно наличие бита проверки на чётность (на рисунке отсутствует), а затем один или два стоповых бита, завершающих передачу пакета и устанавливающих уровень линии в единицу до прихода следующего стартового бита (следующего пакета данных). Структура пакета, формируемого передатчиком, должна

быть известна приёмнику, иначе он не сможет правильно организовать приём.

Другая важная характеристика – это скорость передачи данных. Она должна быть одинаковой для передатчика и приёмника. Скорость последовательной передачи измеряется в бодах. Боды – это количество бит (как информационных так и вспомогательных), передаваемых за секунду.

Последовательный порт персонального компьютера поддерживает следующие скорости передачи данных: 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600 и 115200 бод.

Главная трудность передачи данных через последовательный порт связана с эффектом накапливания погрешности рассинхронизации с течением времени. Он иллюстрируется рисунком, где изображены передаваемые контроллером данные. Жирными точками отмечены моменты возможного изменения сигнала.

Рис. 4. Проблема рассинхронизации

Сигналы синхронизации последовательного порта должны совпадать с серединой каждого бита информации. Пунктиром показаны идеальные синхроимпульсы.

Видно, что в середине стартового бита включается генератор синхронизирующих сигналов приёмника (COM-порта). На каждом синхросигнале считывается значение с входа порта. Если длительность передачи контроллером одного бита пакета отличается от периода синхросигналов, появляется погрешность. С каждым следующим передаваемым битом погрешность накапливается и возможна ошибка. Например, на рисунке 5-й синхроимпульс приходится на 6-й бит пакета.

Аппаратный модуль UART

В состав большинства контроллеров AVR входит модуль UART, который позволяет принимать и получать байты данных автоматически. Управление модулями осуществляется через специальные регистры.

Основным регистром для общения с модулем UART является *UDR* (UART Data Register). На самом деле это два разных регистра, имеющих один адрес. При записи данные попадает в один регистр (регистр передатчика), а при чтении данные вычитываются из другого регистра (регистр приёмника).

Когда байт пришёл в регистр UDR, может сработать прерывание по завершению приёма (которое перед этим нужно разрешить), которое вызывается сразу же, как только приёмник получил все биты данных.

Поскольку передача во времени происходит не мгновенно, то перед записью очередного байта нужно дождаться окончания передачи предыдущего. О том, что *UDR* пуст и готов к приёму нового байта, сигнализирует бит *UDRE*, он же вызовет аппаратное прерывание по опустошению буфера.

Таким образом, всё управление UART можно осуществлять либо при помощи прерываний (что предпочтительнее), либо в цикле проверять значения управляющих регистров.

Настройка модуля UART

Все настройки приёмопередатчика хранятся в регистрах конфигурации. Это *UCSRA, UCSRB* и *UCSRC*. Скорость задаётся в паре регистров *UBBRH:UBBRL*.

7	6	5	4	3	2	1	0
RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
R	R/W	R	R	R	R	R/W	R/W

Рис. 5. Регистр UCSRA (USART Control and Status Register A)

Регистр *UCSRA* содержит биты *RXC* и *TXC*, это флаги завершения приёма и передачи соответственно. RXC устанавливается, когда непрочитанный байт находится в регистре UDR (заполнение регистра приёмника), а TXC устанавливается, когда последний стоп-бит прошёл, а новое значение в UDR не поступило (опустошение регистра передатчика). Также одновременно с этими флагами вызывается прерывание (если оно было разрешено). Сбрасываются биты RXC и TXC аппаратно – принимающий после чтения из регистра UDR, передающий при переходе на прерывание, либо программно (чтобы сбросить флаг программно, в него надо записать единицу).

Бит *UDRE* сигнализирует о том, что регистр UDR передатчика пуст и в него можно записать новый байт. Сбрасывается он аппаратно после засылки в UDR какого-либо байта. При опустошении регистра передатчика может генерироваться прерывание.

Бит *U2X* – бит удвоения скорости при работе в асинхронном режиме. Его надо учитывать при расчёте значения в *UBBRH:UBBRL*

7	6	5	4	3	2	1	0
RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W

Рис. 6. Регистр UCSRB (USART Control and Status Register B)

Регистр *UCSRB* содержит биты *RXEN* и *TXEN* – разрешение приёма и передачи. Для работы UART их нужно установить, т.к. при их сбросе выводы UART работают как обычные пины ввода-вывода.

Биты *RXCIE, TXCIE, UDRIE* разрешают прерывания по завершению приёма, передачи и опустошению буфера передачи UDR.

7	6	5	4	3	2	1	0
URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Рис. 7. Регистр UCSRC (USART Control and Status Register C)

Регистр *UCSRC* в разных моделях AVR может быть организован по-разному. В общем случае регистр UCSRC задаёт количество стоп-битов, наличие бита чётности и способ его проверки. Если оставить все по умолчанию, то получится стандартный режим – 1 стоп-бит, без бита чётности. Надо только задать количество бит в посылке (от 5 до 9). Делается это *битами UCSZ0, UCSZ1, UCSZ2* (бит UCSZ2 находится в регистре UCSRB). Для стандартного формата пакета в 8 бит нужно установить комбинацию UCSZ2 UCSZ1 UCSZ0 в значение 011.

Скорость передачи и приёма задаётся в регистровой паре *UBBRx*. Вычисляется требуемое значение этой регистровой пары по формуле:

UBBR=*XTAL*/(16*baudrate)-1 для U2X=0

UBBR=*XTAL*/(8*baudrate)-1 для U2X=1

Интерфейс I2C

I_2C – двухпроводной интерфейс, разработанный и запатентованный корпорацией Philips. Последняя версия стандарта представлена в 2001 году. С 2006 года за использование интерфейса не требуется платить патентные отчисления, что, вкупе с простотой реализации делает протокол популярным для связи с периферийными устройствами.

Основные термины:

– *Передатчик* – устройство, передающее данные по шине;

– *Приёмник* – устройство, получающее данные с шины;

r (ведущий) – устройство, которое инициирует передачу и формирует тактовый сигнал;

(ведомый) – устройство, к которому обращается «Master»;

– режим работы шины I2C с более чем одним «Master»;

– *Арбитраж* – процедура, гарантирующая, что только один «Master» управляет шиной;

– *Синхронизация* – процедура синхронизации тактового сигнала от двух или более устройств.

Интерфейс I2C. Физический уровень.

Данные передаются по двум проводам – линия данных (SDA) и линия синхронизации (SCL). Всего на одной двухпроводной шине может быть до 127 устройств. Такты генерирует master.

Передача сигналов осуществляется установкой линии в ноль, в единицу линия возвращается сама. Линии подключены к Vcc через подтягивающие резисторы (рекомендуется использовать резисторы номиналом в 10 кОм). Чем больше сопротивление резистора, тем дольше линия восстанавливается в единицу и тем сильней заваливаются фронты импульсов, и падает скорость передачи.

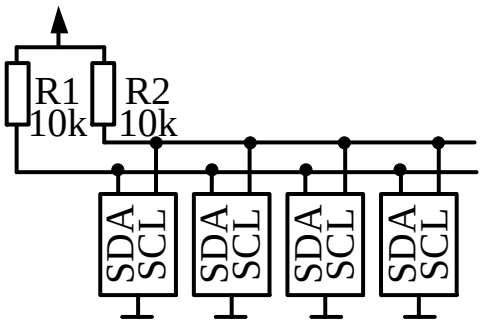


Рис. 8. Схема подключения устройств

Интерфейс I2C. Начало передачи

Процедура обмена начинается с того, что ведущий формирует состояние СТАРТ – ведущий генерирует переход сигнала линии SDA из единицы в ноль при единице на линии SCL. Этот переход воспринимается всеми устройствами, подключёнными к шине как признак начала процедуры обмена.

Генерация синхросигнала – это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

Процедура обмена завершается тем, что ведущий формирует состояние СТОП – переход состояния линии SDA из нуля в единицу при единице линии SCL.

Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.

При передаче посылок по шине I_2C каждый ведущий генерирует свой синхросигнал на линии SCL. После формирования состояния СТАРТ ведущий переводит состояние линии SCL в ноль и выставляет на линию SDA старший бит первого байта сообщения. Количество байт в сообщении не ограничено.

Спецификация шины I_2C разрешает изменения на линии SDA только при нуле на линии SCL. Данные действительны и должны оставаться стабильными только во время состояния единицы на линии SCL.

Для подтверждения приёма байта от ведущего – передатчика ведомым – приёмником в спецификации протокола обмена по шине I_2C вводится специальный бит подтверждения, выставляемый на шину SDA после приёма 8 бита данных.

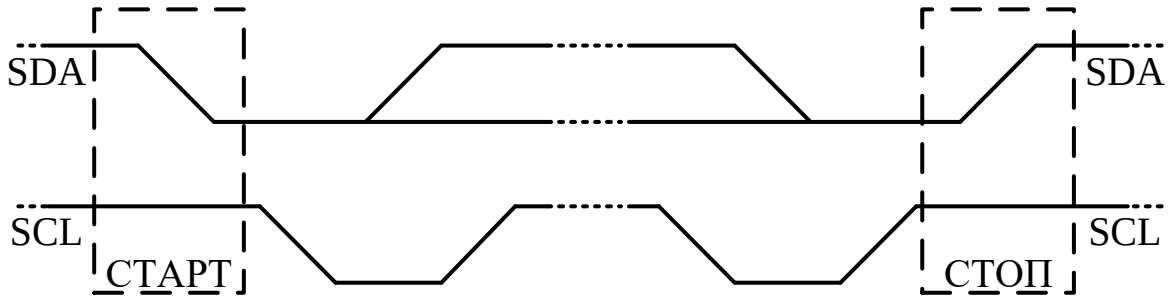


Рис. 9. Сигналы СТАРТ и СТОП

Интерфейс I2C. Подтверждение (ACK)

Передача 8 бит данных от передатчика к приёмнику завершается дополнительным циклом (формированием 9-го тактового импульса линии SCL), при котором приёмник выставляет нулевой уровень сигнала на линии SDA, как признак успешного приёма байта.

Подтверждение при передаче данных обязательно. Соответствующий импульс синхронизации генерируется ведущим. Передатчик отпускает линию SDA (переводит в единицу) на время синхроимпульса подтверждения. Приёмник должен удерживать линию SDA в течение единичного уровня синхроимпульса подтверждения в стабильном нулевом уровне состоянии.

В том случае, когда ведомый-приёмник не может подтвердить свой адрес (например, когда он выполняет в данный момент какие-либо функции реального времени), на линии данных должна быть выставлена единица. После этого ведущий может выдать сигнал СТОП для прерывания пересылки данных.

Если в пересылке участвует ведущий-приёмник, то он должен сообщить об окончании передачи ведомому-передатчику путём не подтверждения последнего байта. Ведомый-передатчик должен освободить линию данных для того, чтобы позволить ведущему выдать сигнал СТОП или повторить сигнал СТАРТ.

Адресация в шине I_2C

Каждое устройство, подключённое к шине, может быть программно адресовано по уникальному адресу. **Для микросхемы внешней EEPROM на плате easyAVR используется адрес 0x51**.

Для выбора приёмника сообщения ведущий использует уникальную адресную компоненту в формате посылки. При использовании однотипных устройств, ИС часто имеют дополнительный селектор адреса, который может быть реализован как в виде дополнительных цифровых входов селектора адреса, так и в виде аналогового входа. При этом адреса таких однотипных устройств оказываются разнесены в адресном пространстве устройств, подключённых к шине. В обычном режиме используется 7-битная адресация.

Процедура адресации на шине I_2C заключается в том, что первый байт после сигнала СТАРТ определяет, какой ведомый адресуется ведущим для

проведения цикла обмена. Исключение составляет адрес «Общего вызова», который адресует все устройства на шине. Когда используется этот адрес, все устройства в теории должны послать сигнал подтверждения. Однако устройства, которые могут обрабатывать «общий вызов», на практике встречаются редко.

Первые семь битов первого байта образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных. «Ноль» означает, что ведущий будет записывать информацию в выбранного ведомого. «Единица» означает, что ведущий будет считывать информацию из ведомого.

После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схема алгоритма работы программы.
3. Комментированный листинг программы на языке ассемблера.
4. Ответы на контрольные вопросы.

Контрольные вопросы

1. Укажите назначение конфигурационных регистров модуля UART?
2. Какими способами можно осуществлять последовательную передачу данных в микроконтроллерах AVR?
3. Какие события генерируются модулем UART?
4. Каков размер адресного пространства шины I2C?
5. В каком режиме осуществляется связь по шине I2C, каковы могут быть роли устройств на шине?

Варианты заданий

Номер варианта	Задание
1	Настройки UART-интерфейса: скорость 4800 бод, контроль чётности – even, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия:

<table>
<tr><td></td><td colspan="2">1. хранить в ОЗУ МК (в переменных типа uint8) переменные {a, b, c, d, e}. После запуска МК до ввода новых значений в качестве значений переменных устанавливается ноль;
2. обеспечивать постоянное отображение значения одной из переменных на четырёх семисегментных индикаторах в виде «<имя_переменной>.<значение>», например, «a.123»;
3. Выполнять обработку внешних прерываний:
3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: a→b→c→d→e→a→…) переменной на семисегментном индикаторе;
3.2. при срабатывании прерывания INT1 выполнять передачу значения переменной, отображаемой в данный момент на семисегментном индикаторе, в терминал ПК (по UART-интерфейсу) в виде «<имя_переменной>=<значение>», например, «a=123».
4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC):
4.1. ввод новых значений для переменных. Формат ввода: «<имя_переменной>=<значение>», например, «a=123»;</td></tr>
<tr><td></td><td>а</td><td>4.2. выполнение арифметических операций {+, -, *} с переменными {a, b, c, d, e}. В ответ МК отправляет результат операции. Формат ввода:
«=<имя_переменной><оператор><имя_переменной>», например, «=a+b»;</td></tr>
<tr><td></td><td>б</td><td>4.3. выполнение сдвиговых и логических операций {>>, <<, |, &} с переменными {a, b, c, d, e}. В ответ МК отправляет результат операции. Формат ввода:
«=<имя_переменной><оператор><имя_переменной>», например, «=a<<c»;</td></tr>
<tr><td></td><td>в</td><td>3.3. выполнять передачу значений всех переменных в терминал ПК (по UART-интерфейсу) при срабатывании прерывания INT0 и нажатой кнопке PD0 в виде «<имя_переменной>=<значение>,…», например, «a=123,b=456,c=789,d=159».</td></tr>
<tr><td>2</td><td colspan="2">Настройки UART-интерфейса: скорость 9600 бод, контроль чётности – odd, один стоповый бит, 8 значащих битов в пакете.</td></tr>
</table>

<table>
<tr><td></td><td colspan="2">Программа для МК должна выполнять следующие действия:
1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI):
1.1. ввод строковых переменных {a, b, c, d}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале:
«имя_строковой_переменной=значение»;
1.2. ввод 8-разрядных неотрицательных целочисленных переменных {x, y}. Формат запроса в терминале:
«имя_целочисленной_переменной=значение»;
2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК:
2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале:
«=имя_переменной»;
2.2. вывод результата операции со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «=название_команды(параметры)».
3. выполнение операции со строковыми переменными, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале:
«имя_переменной=название_команды(параметры)».
В качестве параметров могут выступать только целочисленные и строковые переменные;</td></tr>
<tr><td></td><td>а</td><td>Реализуемые операции:
- x=strcmp(s1,s2) – сравнение строк s1 и s2, результат 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2.
- s0=left(s1,x) – копирование первых x элементов строки s1 в переменную s0</td></tr>
<tr><td></td><td>б</td><td>Реализуемые операции:</td></tr>
</table>

		- x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=xor(s1,x) – запись в переменную s0 результата поразрядного исключающего ИЛИ всех байтов строки s1 с числом x;
	в	Реализуемые *операции*: - x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число); - s0=substr(s1,x,y) – копирование фрагмента строки s1, начинающегося с позиции x и имеющего длину y, в переменную s0;
3	Настройки UART-интерфейса: скорость 28800 бод, контроль чётности выключен, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 16-разрядных (от -2^{15} до 2^{15}) целочисленных переменных $\{a, b, c\}$ в шестнадцатеричной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;	
	а	1.2. ввод логических выражений $\{@, \#, \$\}$, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов $\{+, -, *\}$. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;
	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти

<table>
<tr><td></td><td></td><td>непосредственно перед вычислением). Формат запроса в терминале «=имя_переменной-выражения».</td></tr>
<tr><td>4</td><td colspan="2">Настройки UART-интерфейса: скорость 38400 бод, контроль чётности – even, один стоповый бит, 8 значащих битов в пакете.
Программа для МК должна выполнять следующие действия:
1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI):
1.1. ввод строковых переменных {@, #, $}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале:
«имя_строковой_переменной=значение»;
1.2. ввод 8-разрядных неотрицательных целочисленных переменных {a, b, c}. Формат запроса в терминале:
«имя_целочисленной_переменной=значение»;
2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК:
2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале:
«=имя_переменной»;
2.2. вывод результата операции со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «=название_команды(параметры)».
3. выполнение операции со строковыми переменными, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале:
«имя_переменной=название_команды(параметры)».
В качестве параметров могут выступать только целочисленные и строковые переменные;</td></tr>
<tr><td></td><td>а</td><td>Реализуемые операции:
- x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число);</td></tr>
</table>

		- s0=replace(s1,s2,s3) – копирование строки s1 в переменную s0, с заменой всех вхождений подстроки s2 на подстроку s3. Если при замене получается строка длиннее 20 символов, то остаются только первые 20 символов;
	б	Реализуемые *операции*: - x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=inv(s1) – копирование инвертированной строки s1 в переменную s0. Пример: inv(“123abc”) = “cba321”;
	в	Реализуемые *операции*: - x=strcmp(s1,s2) – сравнение строк s1 и s2 – 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2; - s0=trim(s1) – копирование строки s1 в переменную s0, предварительно удаляя все пробелы (код 0x32) в начале и конце строки;
5	Настройки UART-интерфейса: скорость 57600 бод, контроль чётности – odd, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 8-разрядных (от -2^7 до 2^7) целочисленных переменных {*x, y, z*} в шестнадцатеричной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;	
	а	1.2. ввод логических выражений {*A, B, C, D*}, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов {& – *побитовое AND*, | – *побитовое OR*, ^ – *побитовое XOR*, ~ – *инверсия*}. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;

	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти непосредственно перед вычислением). Формат запроса в терминале «*=имя_переменной-выражения*».
6	Настройки UART-интерфейса: скорость 4800 бод, контроль чётности выключен, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменных типа unsigned char *) строк {*x, y, z*}, состоящих из комбинаций следующих символов {<пробел>,0,1,2,3,4,5,6,7,8,9, a,b,c,d,e,f}, длины строк не должны превышать 20 символов. После запуска МК до ввода значений в качестве значения {*x, y, z*} устанавливаются пустые строки; 2. обеспечивать постоянное отображение одной из переменных {*x, y, z*} в виде бегущей строки на четырёх семисегментных индикаторах. При отображении переменной *x* должен светиться светодиод PD7, при отображении переменной *y* – PD6, *z* – PD5; 3. Выполнять обработку внешних прерываний: 3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: *x*→*y*→*z*→*x*→…) переменной на семисегментном индикаторе. 3.2. при срабатывании прерывания INT1 выполнять передачу значения переменной, отображаемой в данный момент на семисегментном индикаторе, в терминал ПК (по UART-интерфейсу) в виде «*<имя_переменной>=<значение>*», например, «*x*=123». 4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 4.1. считывать новое значение переменных {*x, y, z*} по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC) в виде «*<имя_переменной>=<строка>*», например, «x=12 34 56 78 90 ab cd ef».	

	а	4.2. считывать строку *a*, которая будет дописана в конец текущей строки *s*: *s = strcat(s, a)*. Если при объединении получается строка длиннее 20 символов, то в переменную *s* сохраняются только первые 20 символов. Формат ввода: «+<*дописываемая строка*>», например, «+extra_string»;
	б	3.3. при срабатывании прерывания INT0 и зажатой кнопке PD0 изменить направление движения бегущей строки
	в	3.4. при срабатывании прерывания INT1 и зажатой кнопке PD0 очистить содержимое текущей строки
7	Настройки UART-интерфейса: скорость 9600 бод, контроль чётности – even, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменных типа uint16) переменные {*a, b, c*}. После запуска МК до ввода новых значений в качестве значений переменных устанавливается ноль; 2. обеспечивать постоянное отображение значения одной из переменных в шестнадцатеричной системе счисления на четырёх семисегментных индикаторах в виде: «<*имя_переменной*>.<*значение*>», например, «a. F2»; 3. Выполнять обработку внешних прерываний: 3.1. при срабатывании прерывания INT0 переходить к выводу следующей (циклически: *a→b→c→a→…*) переменной на семисегментном индикаторе; 3.2. при срабатывании прерывания INT1 сбросить значение всех переменных в ноль. 4. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 4.1. ввод новых значений для переменных. Формат ввода: «<*имя_переменной*>=<*значение*>», например, «a=123»;	
	а	3.3. выполнять передачу значений всех переменных в терминал ПК (по UART-интерфейсу) при срабатывании прерывания INT0 и зажатой кнопке PD0 в виде «<*имя_переменной*>=<*значение*>,…», например, «a=123,b=456,c=789,d=159».

	б	4.2. выполнение арифметических операций {+, -} с переменными {*a, b, c*}. В ответ МК отправляет результат операции. Формат ввода: «=<*имя_переменной*><*оператор*><*имя_переменной*>», например, «=a+b»;
	в	4.3. выполнение сдвиговых и логических операций {|, &, ^} с переменными {*a, b, c, d, e*}. В ответ МК отправляет результат операции. Формат ввода: «=<*имя_переменной*><*оператор*><*имя_переменной*>», например, «=a<<c»;
8	Настройки UART-интерфейса: скорость 28800 бод, контроль чётности – odd, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. хранить в ОЗУ МК (в переменной типа unsigned char *) строку *s*, состоящую из печатных ASCII-символов (коды символов должны быть больше 31), длина строки не должна превышать 100 символов. После запуска МК до ввода новой строки в качестве значения *s* устанавливается пустая строка; 2. Выполнять обработку внешних прерываний: 2.1. при срабатывании прерывания INT0 выводить строку *s* в терминал ПК (по UART-интерфейсу); 2.2. при срабатывании прерывания INT1 выполнять очистку строки *s*; 3. выполнять следующие команды, поступающие по UART-интерфейсу с терминала ПК (для считывания команд необходимо использовать прерывание UART_RXC): 3.1. считывать новое значение строки *s*. Формат ввода в терминал: «*<*новая строка*>», например, «*new_string»;	
	а	3.2. считывать целое беззнаковое число *y*, для выполнения поразрядного исключающего ИЛИ с каждым байтом строки *s* с сохранением результата в переменную *s*. Формат ввода: «^*number*», например, «^26»; 3.3. считывать строку *x*, которая будет дописана в конец текущей строки *s*: *s = strcat(s, x)*. Если при объединении получается строка длиннее 100 символов, то в переменную *s* сохраняются только первые 100 символов. Формат ввода: «+<*дописываемая строка*>», например, «+extra_string»;

	б	3.4. считывать строку *x*, для который будет выполнен поиск всех вхождений подстроки *x* в строке *s* с удалением подстроки и схлопыванием строки *s*. Формат ввода: «-<*удаляемая_строка*>», например, «-delete_me»; 3.5. считывать целое беззнаковое число *y*, показывающее номер символа строки *s*, значение которого необходимо вернуть в терминал в виде цифр в шестнадцатеричном представлении – *0xXX*. Формат ввода: «*?number*», например, «?26»;
	в	3.6. считывать целое беззнаковое число *y*, показывающее количество символов, соответствующее новой длине переменной *s*, все символы за пределами обновляемой строки заменяются нулём, все символы, добавляемые к строке, заполняются числом 0x32 (пробелом). Формат ввода: «*/number*», например, «/26»; 3.7. считывать строку *x*, которая будет использоваться в качестве подстроки при поиске в переменной *s*. Если подстрока *x* найдена, то в терминал возвращается номер первого символа первого вхождения подстроки в строке, если не найдена – число *0xFF*. Формат ввода: «*#<подстрока>*», например, «#find_substring»;
9		Настройки UART-интерфейса: скорость 38400 бод, контроль чётности выключен, два стоповых бита, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод 16-разрядных (от -2^{15} до 2^{15}) целочисленных переменных $\{a, b, c, d, e\}$ в десятичной системе счисления. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных переменных. Формат запроса в терминале: «*=имя_целочисленной_переменной*»;
	а	1.2. ввод логических выражений $\{X, Y, Z\}$, состоящих из целочисленных переменных из прошлого пункта, 16-разрядных целочисленных констант и операторов {lr – логический сдвиг

		вправо, ll – логический сдвиг влево, rr – циклический сдвиг вправо, rl – циклический сдвиг влево, & – *AND*, | – *OR* }. Длина выражения не может быть больше 15 символов. Формат запроса в терминале: «*имя_переменной-выражения=значение*»;
	б	2.2. вывод формулы переменных-выражений. Формат запроса в терминале: «*?имя_переменной-выражения*»;
	в	2.3. вывод значения вычисления по формулам переменных-выражений (значения задействованных в формуле целочисленных переменных считываются из внешней памяти непосредственно перед вычислением). Формат запроса в терминале «*=имя_переменной-выражения*».
10		Настройки UART-интерфейса: скорость 57600 бод, контроль чётности – even, один стоповый бит, 8 значащих битов в пакете. Программа для МК должна выполнять следующие действия: 1. ввод данных с терминала ПК (передача на МК по UART-интерфейсу) и сохранение их во внешней памяти EEPROM (интерфейс TWI): 1.1. ввод строковых переменных {*x*, *y*, *z*}. Вводимая строка может содержать печатные ASCII-символы (коды символов должны быть больше 31), длина строки не должна превышать 20 символов. Формат запроса в терминале: «*имя_строковой_переменной=значение*»; 1.2. ввод 8-разрядных неотрицательных целочисленных переменных {*A*, *B*, *C*}. Формат запроса в терминале: «*имя_целочисленной_переменной=значение*»; 2. Запрос информации от МК (в том числе из подключённой к нему внешней памяти EEPROM) с выводом в терминал ПК: 2.1. вывод значений целочисленных и строковых переменных на терминал ПК. Формат запроса в терминале: «*=имя_переменной*»; 2.2. вывод результата *операции* со строковыми переменными на терминал ПК, значения задействованных в операции целочисленных и строковых переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале «*=название_команды(параметры)*». 3. выполнение *операции* со строковыми переменными, значения задействованных в операции целочисленных и строковых

	переменных считываются из внешней памяти непосредственно перед выполнением операции. Формат запроса в терминале: «*имя_переменной=название_команды(параметры)*». В качестве параметров могут выступать только целочисленные и строковые переменные;	
	а	Реализуемые *операции*: - x=find(s1,s2) – вычисление номера первого символа первого вхождения подстроки s2 в строке s1, если подстрока s2 в строке s1 отсутствует – 0xFF; - s0=xor(s1,x) – запись в переменную s0 результата поразрядного исключающего ИЛИ всех байтов строки s1 с числом x;
	б	Реализуемые *операции*: - x=strcmp(s1,s2) – сравнение строк s1 и s2 – 8-разрядное беззнаковое число: 0x00, если s1 = s2; 0x01, если s1 > s2; 0xFF, если s1 < s2; - s0=strcat(s1,s2) – объединение строк s1 и s2 и помещение результата в переменную s0. Если при объединении получается строка длиннее 20 символов, то остаются только первые 20 символов;
	в	Реализуемые *операции*: - x=strlen(s1) – вычисление длины строки s1 (8-разрядное беззнаковое число); - s0=strcpy(s1) – копирование строки s1 в переменную s0;

Лабораторная работа 7

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА И ПЭВМ. ПРОТОТИП ЗАЩИЩЁННОГО ХРАНИЛИЩА ДАННЫХ

Цель работы

Получение практических навыков по организации двунаправленного взаимодействия между микроконтроллером и персональным компьютером. Знакомство с основами построения файловых систем.

Порядок выполнения работы

1. Разработать программно-аппаратный прототип внешнего хранилища данных. В качестве места хранения данных выступает внутренняя память данных МК ATmega32 (EEPROM). Формат хранения данных соответствует упрощённой структуре файловой системы FAT. В качестве хранимых объектов могут выступать файлы и каталоги, каталоги образуют иерархическую структуру. Для каждого файла/каталога отводится набор кластеров (но не менее одного кластера на объект).

Массив памяти (1 Кбайт) разделяется на 32 кластера по 32 байта каждый. В первом кластере размещается ***массив указателей*** (длина каждого указателя равна 1 байту) – цепочка кластеров, описывающих содержимое каждого кластера памяти (номер указателя в массиве соответствует номеру кластера в памяти; массив из 32 однобайтных элементов соответствует 32 кластерам памяти). Записи в массиве указателей содержат следующую информацию:

– если в кластере содержится не последний фрагмент файла/каталога – номер кластера, в котором хранится следующий фрагмент файла;

– если в кластере содержится последний фрагмент файла/каталога, то запись содержит число 0x20;

– если кластер не занят, то запись содержит число 0;

– если файл/каталог, размещённый в кластере, был удалён, то запись содержит число 0x40.

В первом байте массива указателей размещается значение – количество свободных в данный момент кластеров и 3 системных бита, указывающих тип файловой системы.

На рисунке 1 представлен пример фрагмента массива указателей.

	7	6	5	4	3	2	1	0	Примечание
0	*Тип ФС*			1	1	1	0	0	- количество свободных кластеров
1	0	0	1	0	0	0	0	0	- корневой каталог
2	0	0	0	0	0	1	0	0	- начало второго объекта
3	0	1	0	0	0	0	0	0	- третий объект (удалён)
4	0	0	1	0	0	0	0	0	- конец второго объекта
5	0	0	0	0	0	0	0	0	- пустой кластер

Рис. 1. Пример массива указателей. Корневой каталог содержится в кластере 1, второй в кластерах 2 и 4, третий объект, находившийся в кластере 3, был удалён. Свободных кластеров – 28 (0x1C).

Во втором кластере размещается корневой каталог, имеющий структуру, идентичную любому другому каталогу.

Информация о файлах и каталогах содержится в виде 4-байтных дескрипторов, размещённых подряд в кластере(ах), описывающих родительский каталог. Примерный вариант содержимого дескриптора приведён на рисунке 2.

	7	6	5	4	3	2	1	0
0	Имя файла/каталога – старший байт							
1	Имя файла/каталога – младший байт							
2	Номер первого кластера						Тип	R	U
3	Размер файла/каталога							

Рис. 2. Пример дескриптора файла/каталога

В качестве имени файла или каталога может выступать пара любых ASCII символов, за исключением следующих: {0x00-0x19, «/», «\», «:», «?», «*», «.»}. Поле «Тип» в дескрипторе содержит «1», если объект является файлом, иначе «0». Поле «R» устанавливается в 1, если вложенный объект доступен для изменения только Администратору. Поле U устанавливается в единицу, если объект доступен только Пользователю. Одновременная установка битов R и U в 0 означает общедоступный объект, одновременная установка битов R и U в 1 запрещена. Размер файла/каталога хранит величину файла в байтах, для каталога это поле содержит значение 0.

Для осуществления операций к защищённым файлам необходимо предварительно выполнить процедуру входа с соответствующим уровнем доступа и ввести pin-код.

При создании каталога в первый дескриптор помещается запись с именем «..», содержащая номер первого кластера родительского каталога с нулевыми остальными полями. Для корневого каталога такой записи не

должно быть. В родительском каталоге должны находиться два предопределённых файла с именами RP и UP, доступные соответственно Администратору и Пользователю и содержащие их pin-коды. Возможные размеры pin-кода уточняются в варианте задания.

В МК должна осуществляться обработка набора команд от ПК (в соответствии с вариантом задания) по управлению файлами и каталогами.

Взаимодействие между устройствами должно осуществляться по интерфейсу UART через виртуальный COM-порт. Протокол передачи данных между персональным компьютером и микроконтроллером разрабатывается самостоятельно.

Приложение для ПК должно позволять выполнять заданное множество команд по работе с файлами, требования к внешнему интерфейсу не предъявляются.

2. Скомпилировать и записать программу в контроллер.

3. Проверить правильность работы программного комплекса.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).

2. Блок-схемы алгоритмов программ для микроконтроллера и ПК.

3. Комментированные листинги программы для микроконтроллера на языках Си и ассемблер и листинг программы для ПК.

4. Описание форматов пересылаемых наборов данных между микроконтроллером и персональным компьютером.

5. Выводы по работе.

Варианты заданий

Номер варианта	Задание
1	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется в ПК, длина pin-кода 1-8 байт. 2. На семисегментном индикаторе постоянно отображается имя текущего каталога (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);

<table>
<tr><td></td><td colspan="2">4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);
4.3. Logout – завершение текущего сеанса;
4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим;
4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим;
4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой;
4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой;
4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой;
5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36</td></tr>
<tr><td></td><td>а</td><td>4.9. DeleteDirectory – удалить указанный каталог;
4.10. GetFileInfo – получить информацию о размере и времени создания указанного файла;</td></tr>
<tr><td></td><td>б</td><td>4.11. GetNextFile – найти следующий файл в текущем каталоге и вернуть его имя;
4.12. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля;</td></tr>
<tr><td></td><td>в</td><td>4.13. DeleteFile – удалить указанный файл;
4.14. GetDirectoryInfo – получить информацию о количестве и общем объёме вложенных файлов (включая все уровни вложенных каталогов) и времени создания указанного каталога;</td></tr>
<tr><td>2</td><td colspan="2">Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия:
1. Ввод pin-кода осуществляется на отладочной плате с регистра PORTB, длина pin-кода 1 байт.
2. На семисегментном индикаторе постоянно отображается имя последнего открытого файла (в шестнадцатеричном формате).</td></tr>
</table>

		3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36
	а	4.9. SetRootPin – настроить pin-код администратора (команда доступна только администратору); 4.10. GetNextDirectory – найти следующий каталог в текущем каталоге и вернуть его имя;
	б	4.11. GetDirectoryInfo – получить информацию о количестве и общем объёме вложенных файлов (включая все уровни вложенных каталогов) и времени создания указанного каталога; 4.12. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора;
	в	4.13. DeleteFile – удалить указанный файл;

		4.14. GetFileInfo – получить информацию о размере и времени создания указанного файла;
3	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется в ПК, длина pin-кода 1-5 байт. 2. На семисегментном индикаторе постоянно отображается код последней выполненной команды файловой системы (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. При запуске МК должна выполняться проверка корректности массива указателей файловой системы, если обнаружены ошибки, то взаимодействие с МК блокируется, а на семисегментном индикаторе постоянно отображается сообщении «Err».	
	а	4.9. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора; 4.10. GetNextDirectory – найти следующий каталог в текущем каталоге и вернуть его имя;
	б	4.11. DeleteFile – удалить указанный файл;

		4.12. FindDirectory – выполнить поиск каталога по указанному короткому имени (2 байта), вернуть путь до всех найденных каталогов. При выполнении команды должна осуществляться проверка доступности каталога текущему пользователю;
	в	4.13. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля; 4.14. GetNextFile – найти следующий файл в текущем каталоге и вернуть его имя;
4	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется на отладочной плате с регистра PORTB, длина pin-кода 1 байт. 2. На семисегментном индикаторе постоянно отображается имя текущего каталога (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; При запуске МК должна выполняться проверка корректности полей всех дескрипторов файловой системы, если обнаружены ошибки, то взаимодействие с МК блокируется, а на семисегментном индикаторе постоянно отображается сообщении «Err».	
	а	4.9. FindFile – выполнить поиск файла по указанному короткому имени (2 байта), вернуть путь до всех найденных файлов. При выполнении команды должна осуществляться проверка доступности файла и каталога текущему пользователю; 4.10. DeleteDirectory – удалить указанный каталог;
	б	4.11. SetRootPin – настроить pin-код администратора (команда доступна только администратору); 4.12. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора;
	в	4.13. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля; 4.14. DeleteFile – удалить указанный файл;

<table>
<tr><td>5</td><td>Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия:
1. Ввод pin-кода осуществляется в ПК, длина pin-кода 4 байта.
2. На семисегментном индикаторе постоянно отображается имя последнего открытого файла (в шестнадцатеричном формате).
3. При поступлении на МК внешних прерываний выполняется следующее:
3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе.
4. выполнять следующие команды файловой системы:
4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);
4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);
4.3. Logout – завершение текущего сеанса;
4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим;
4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим;
4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой;
4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой;
4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой;
5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36</td></tr>
<tr><td>а</td><td>4.9. FindDirectory – выполнить поиск каталога по указанному короткому имени (2 байта), вернуть путь до всех найденных каталогов. При выполнении команды должна осуществляться проверка доступности каталога текущему пользователю;</td></tr>
</table>

		4.10. GetDirectoryInfo – получить информацию о количестве и общем объёме вложенных файлов (включая все уровни вложенных каталогов) и времени создания указанного каталога;
	б	4.11. GetNextFile – найти следующий файл в текущем каталоге и вернуть его имя; 4.12. SetUserPin – настроить pin-код пользователя (команда доступна только пользователю);
	в	4.13. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора; 4.14. GetFileInfo – получить информацию о размере и времени создания указанного файла;
6	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется на отладочной плате с регистра PORTB, длина pin-кода 1 байт. 2. На семисегментном индикаторе постоянно отображается код последней выполненной команды файловой системы (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой;	

<table>
<tr><td></td><td colspan="2">4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой;
5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36</td></tr>
<tr><td></td><td>а</td><td>4.9. GetFileInfo – получить информацию о размере и времени создания указанного файла;
4.10. DeleteDirectory – удалить указанный каталог;</td></tr>
<tr><td></td><td>б</td><td>4.11. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора;
4.12. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля;</td></tr>
<tr><td></td><td>в</td><td>4.13. DeleteFile – удалить указанный файл;
4.14. GetNextFile – найти следующий файл в текущем каталоге и вернуть его имя;</td></tr>
<tr><td>7</td><td colspan="2">Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия:
1. Ввод pin-кода осуществляется в ПК, длина pin-кода 3 байта.
2. На семисегментном индикаторе постоянно отображается имя текущего каталога (в шестнадцатеричном формате).
3. При поступлении на МК внешних прерываний выполняется следующее:
3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе.
4. выполнять следующие команды файловой системы:
4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);
4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);
4.3. Logout – завершение текущего сеанса;
4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим;
4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет</td></tr>
</table>

	места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. При запуске МК должна выполняться проверка корректности полей всех дескрипторов файловой системы, если обнаружены ошибки, то взаимодействие с МК блокируется, а на семисегментном индикаторе постоянно отображается сообщении «Err».	
	а	4.9. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора; 4.10. DeleteFile – удалить указанный файл;
	б	4.11. SetUserPin – настроить pin-код пользователя (команда доступна только пользователю); 4.12. FindFile – выполнить поиск файла по указанному короткому имени (2 байта), вернуть путь до всех найденных файлов. При выполнении команды должна осуществляться проверка доступности файла и каталога текущему пользователю;
	в	4.13. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля; 4.14. GetNextDirectory – найти следующий каталог в текущем каталоге и вернуть его имя;
8	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется на отладочной плате с регистра PORTB, длина pin-кода 1 байт. 2. На семисегментном индикаторе постоянно отображается имя последнего открытого файла (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода);	

	4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. При запуске МК должна выполняться проверка корректности массива указателей файловой системы, если обнаружены ошибки, то взаимодействие с МК блокируется, а на семисегментном индикаторе постоянно отображается сообщении «Err».	
	а	4.9. SetUserPin – настроить pin-код пользователя (команда доступна только пользователю); 4.10. FindFile – выполнить поиск файла по указанному короткому имени (2 байта), вернуть путь до всех найденных файлов. При выполнении команды должна осуществляться проверка доступности файла и каталога текущему пользователю;
	б	4.11. DeleteFile – удалить указанный файл; 4.12. GetNextFile – найти следующий файл в текущем каталоге и вернуть его имя;
	в	4.13. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора; 4.14. GetNextDirectory – найти следующий каталог в текущем каталоге и вернуть его имя;
9	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется в ПК, длина pin-кода 1-8 байт. 2. На семисегментном индикаторе постоянно отображается код последней выполненной команды файловой системы (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее:	

		3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36
	а	4.9. FindFile – выполнить поиск файла по указанному короткому имени (2 байта), вернуть путь до всех найденных файлов. При выполнении команды должна осуществляться проверка доступности файла и каталога текущему пользователю; 4.10. DeleteFile – удалить указанный файл;
	б	4.11. GetFileInfo – получить информацию о размере и времени создания указанного файла; 4.12. SetRootPin – настроить pin-код администратора (команда доступна только администратору);
	в	4.13. DeleteDirectory – удалить указанный каталог; 4.14. GetNextDirectory – найти следующий каталог в текущем каталоге и вернуть его имя;

10	Программный комплекс состоит из пользовательского интерфейса (ПК) и хранилища данных (МК) и должен выполнять следующие действия: 1. Ввод pin-кода осуществляется на отладочной плате с регистра PORTB, длина pin-кода 1 байт. 2. На семисегментном индикаторе постоянно отображается имя текущего каталога (в шестнадцатеричном формате). 3. При поступлении на МК внешних прерываний выполняется следующее: 3.1. При поступлении прерывания Int0 последовательно отображает имена каталогов от корневого до текущего с задержкой на 0,5 с каждого значения, затем возврат к обычной работе. 4. выполнять следующие команды файловой системы: 4.1. LoginAsRoot – авторизация с правами администратора (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.2. LoginAsUser – авторизация с правами пользователя (если pin-код не указан в файловой системе, то сохранение введённого pin-кода); 4.3. Logout – завершение текущего сеанса; 4.4. CreateDirectory – создание каталога с указанным именем, например, «AA\A1\01», если указанного головного каталога не существует или нет места для размещения нового каталога, команда завершается с ошибкой, если ошибок нет созданный каталог становится текущим; 4.5. CreateFile – создание файла с указанным именем, например, «AA\A1\01\ab», если указанного головного каталога не существует или нет места для размещения нового каталога команда завершается с ошибкой, если ошибок нет головной каталог становится текущим; 4.6. ListDirectory – вернуть список имён объектов (файлов и каталогов) в указанном каталоге, если указанного каталога не существует, команда завершается с ошибкой; 4.7. ReadFile – прочитать содержимое указанного файла, если указанный файл не существует, команда завершается с ошибкой; 4.8. WriteFile – записать переданную в команде последовательность байтов в указанный файл, если указанный файл не существует, команда завершается с ошибкой; 5. Представленный в описании дескриптор необходимо расширить на четыре байта, поместив в них дату и время создания объекта: биты 0-4 – счётчик секунд по 2 (0-29), биты 5-10 – минуты (0-59), биты 11-15 – часы (0-23), биты 16-20 – день месяца (1-31), биты 21-24 – месяц (1-12), биты 25-31 – год от 1980 (0-127), например, 0b0101.0110.0110.1101.0101.0000.0001.0010 соответствует 13 марта 2023 года 10:00:36	
	а	4.9. DeleteFile – удалить указанный файл; 4.10. UnlockUser – разблокировать пользователя (команда доступна только администратору). Блокирование происходит после двух последовательных неправильных вводов пароля;

	б	4.11. GetDirectoryInfo – получить информацию о количестве и общем объёме вложенных файлов (включая все уровни вложенных каталогов) и времени создания указанного каталога; 4.12. SetUserPin – настроить pin-код пользователя (команда доступна только пользователю);
	в	4.13. FormatFS – отформатировать хранилище (команда доступна только администратору), входным параметром является новый pin-код администратора; 4.14. GetFileInfo – получить информацию о размере и времени создания указанного файла;

Лабораторная работа 8

ВЗАИМОДЕЙСТВИЕ МИКРОКОНТРОЛЛЕРА И ПЭВМ. ПРОТОТИП СИСТЕМЫ ШИФРОВАНИЯ

Цель работы

Получение практических навыков по организации двунаправленного взаимодействия между микроконтроллером и персональным компьютером. Знакомство с криптосистемой RSA и криптографическими протоколами Диффи-Хеллмана и Эль-Гамаля.

Теоретические сведения

Криптосистема RSA

Принцип действия криптосистемы RSA заключается в следующем. Пусть p и q – два больших простых числа и пусть $n = pq$. Выбирается число e, $1 < e < \varphi(n) - 1$, взаимно простое со значением функции Эйлера $\varphi(n) = (p - 1)(q - 1)$, и вычисляется d – мультипликативно обратное к e: $d \equiv e^{-1} \pmod{\varphi(n)}$, то есть $ed \equiv 1 \pmod{\varphi(n)}$. Числа e и d называются *открытым* и *секретным показателями* соответственно. Открытым ключом является пара (n, e), секретным ключом – число d. Множители p и q должны храниться в секрете.

Для зашифрования сообщения m нужно вычислить шифртекст $c \equiv m^e \pmod{n}$, для расшифрования – возвести шифртекст в степень d: $m \equiv c^d \pmod{n}$.

Аналогично реализуется цифровая подпись: подписью сообщения m является число $s \equiv m^d \pmod{n}$, где d – секретный ключ формирования подписи.

Протокол установки ключей Диффи-Хеллмана

Схема работы протокола состоит из трёх этапов:

1. Сторона 1: генерирует случайный секретный показатель x и передаёт второй стороне a^x. Также передаются параметры системы: образующая a группы и число p. Таким образом, передаются три параметра (a^x, a, p)

2. Сторона 2: при входящем подключении принимает (a^x, a, p), генерирует случайный показатель y, вычисляет a^y и a^{xy} и передаёт стороне 1 a^y.

3. Сторона 1: принимает a^y, вычисляет a^{xy}.

Далее стороны могут обмениваться зашифрованными сообщениями. При передаче файлов необходимо передавать длину открытого текста, так как только передача длины позволит считать из потока соединения нужное количество байт. Причём передаётся длина открытого текста до зашифрования.

Протокол Эль-Гамаля

Протокол шифрования Эль-Гамаля с открытым ключом основан на задаче Диффи-Хеллмана. В протоколе Диффи-Хеллмана оба показателя x и y случайны. В протоколе Эль-Гамаля один показатель фиксирован и служит секретным ключом расшифрования, а другой показатель является случайным.

Пусть M – множество сообщений и G – группа. Открытым ключом, общим для всех пользователей системы, являются группа G и образующая a. Кроме того, существует общедоступная функция $f: M \times G \to G$, обратимая по первому аргументу. Персональным секретным ключом является показатель x, персональным открытым ключом – экспонента $b = a^x$.

Схема зашифрования сообщения m:

1. Генерирует случайный показатель y.

2. Вычисляет элементы $a^y \pmod p$, $b^y \pmod p \in G$ и значение $c \leftarrow f(m, b^y)$.

Шифрограммой является пара (a^y, c).

Схема расшифрования шифрограммы (a^y, c):

1. Возводит a^y в степень x, получает при этом $a^{xy} \equiv b^y \pmod p$.

2. Находит сообщение $m \leftarrow f^{-1}(c, b^y)$.

Вид функции f зависит от группы G. Если $G = \oplus_p^*$ – мультипликативная группа простого поля, то в качестве f могут использоваться функции $f(m, b^y) = mb^y \pmod p$, $f(m, b^y) = m + b^y \pmod p$, $f(m, b^y) = m - b^y \pmod p$ и т. п.

Порядок выполнения работы

1. Получить вариант задания у преподавателя
2. Разработать программный прототип криптосистемы с учётом дополнительных требований в соответствии с вариантом.

Программный комплекс должен состоять из двух частей: программа для микроконтроллера и приложение для ПК. Программа для МК должна осуществлять хранение параметров криптосистемы (ключей), управление ключами в соответствии с поступающими от ПК командами, шифрование и расшифрование двоичных данных, поступающих с ПК, а также подписание хэша (только для RSA) и проверку подписи хэша (только для RSA), поступающего с ПК. Недопустима предварительная передача всего файла на МК с последующей обработкой и возвратом полностью сформированного зашифрованного/расшифрованного файла – обработка двоичных данных, поступающих с ПК, должна выполняться побайтно (пословно) или блоками (при соответствующем указании в варианте задания).

Приложение для ПК должно предоставлять пользователю интерфейс (допускается разработка консольного приложения) для управления параметрами криптосистемы (считывание параметров из памяти МК, генерация и запись ключей в память МК, удаление отдельных ключей), возможность ввода (выбора) имён файлов для считывания и записи двоичных данных, указания выполняемого действия (шифрование, расшифрование, вычисление хэша и электронной подписи, вычисление хэша и проверка электронной подписи).

Взаимодействие между устройствами должно осуществляться по интерфейсу UART-виртуальный COM-порт. Протокол передачи данных между ПК и МК разрабатывается самостоятельно. В протоколе необходимо предусмотреть передачу информации об ошибках от МК, например, при указании неизвестного открытого ключа в командах, не связанных с установкой ключей.

3. Скомпилировать и записать программу в контроллер.
4. Проверить правильность работы программного комплекса.

Содержание отчёта

1. Схема установки (задействованные узлы отладочной платы).
2. Блок-схемы алгоритмов программ для микроконтроллера и ПК.
3. Комментированные листинги программы для микроконтроллера на языках Си и ассемблер и листинг программы для ПК на языке Си.
4. Описание форматов пересылаемых наборов данных между микроконтроллером и персональным компьютером.
5. Выводы по работе.

Варианты заданий

Номер варианта	Задание
1	Протокол Эль-Гамаля с открытым ключом. При шифровании необходимо использовать функцию $f(m, b^y) = m + b^y \pmod p$. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (a, p, x). При шифровании файла на контроллер должна передаваться пара (a, p), в ответном сообщении вместе с зашифрованным файлом должно возвращаться значение a^y (одно на всё сообщение), допускается в качестве значения y использовать предопределённый набор однобайтных величин. При расшифровании вместе с телом файла на микроконтроллер должна передаваться тройка (a, p, a^y) и микроконтроллер должен выполнять поиск секретного ключа по записям в EEPROM. Каждый набор ключей должен сопровождаться pin-кодом в виде двух 8-разрядных чисел A и B: в EEPROM должны содержаться блоки (a, p, x, A, B). При выборе операции расшифрования или подписания пользователь должен вводить pin-код на портах A и B (одновременно нажать нужное сочетание кнопок). Если в течение пяти секунд нужная комбинация не нажата – на ПК выдаётся сообщение об ошибке. При вводе правильного pin-кода **немедленно** начинается соответствующая операция. В качестве параметров для проверки работоспособности можно использовать следующие величины: $a = 6397$, $p = 8963$ и $x = 7499$.
2	Криптосистема RSA. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (n, e, d). При шифровании, расшифровании, подписании и проверке подписи открытый ключ (n, e) должен передаваться вместе с телом файла (хэшем) и при расшифровании и подписании микроконтроллер должен выполнять поиск секретного ключа по записям открытых ключей в EEPROM.

	МК должен вести подсчёт статистики по количеству команд шифрования, расшифрования по всем существующим в EEPROM ключам, а также единый набор по отсутствующим ключам. При поступлении команды с ПК обновлённая статистика должна сохраняться в EEPROM (информация по использованию ключа должна быть рядом с самим ключом: (n, e, d, c)). В приложении для ПК должна быть возможность получить всю текущую статистику (массив (n, e, c) и число команд с несуществующим ключом) и записать в отдельный файл/вывести в собственное окно, а также должна быть возможность обнуления всей статистики. В качестве параметров для проверки работоспособности можно использовать следующие величины: $n = 39913$, e $= 453$ и $d = 1221$.
3	Протокол Эль-Гамаля с открытым ключом. При шифровании необходимо использовать функцию $f(m, b^y) = m + b^y \pmod p$. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (a, p, x). При шифровании файла на контроллер должна передаваться пара (a, p), в ответном сообщении вместе с зашифрованным файлом должно возвращаться значение a^y (одно на всё сообщение), допускается в качестве значения y использовать предопределённый набор однобайтных величин. При расшифровании вместе с телом файла на микроконтроллер должна передаваться тройка (a, p, a^y) и микроконтроллер должен выполнять поиск секретного ключа по записям в EEPROM. Шифрование и расшифрование файлов должно выполняться блоками переменной длины. Должна быть реализована возможность выбора длины блока из приложения для ПК, длина блока должна храниться в EEPROM (сохраняться при изменении на ПК), а также считываться при запуске приложения на ПК. В качестве параметров для проверки работоспособности можно использовать следующие величины: $a = 6397$, $p = 8963$ и $x = 7499$.
4	Криптосистема RSA. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (n, e, d). При шифровании, расшифровании, подписании и проверке подписи открытый ключ (n, e) должен передаваться вместе с телом файла (хэшем) и при расшифровании и подписании микроконтроллер должен выполнять поиск секретного ключа по записям открытых ключей в EEPROM. При поступлении запроса на изменение параметров криптосистемы МК должен ожидать нажатия определённой комбинации кнопок на портах A-D в течение трёх секунд. Если нужная комбинация не была вовремя нажата запрос игнорируется. При вводе правильной комбинации операция начинает выполняться **немедленно**.

	В качестве параметров для проверки работоспособности можно использовать следующие величины: $n = 39913$, e = 453 и $d = 1221$.
5	Протокол Эль-Гамаля с открытым ключом. При шифровании необходимо использовать функцию $f(m, b^y) = m + b^y \pmod p$. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (a, p, x). При шифровании файла на контроллер должна передаваться пара (a, p), в ответном сообщении вместе с зашифрованным файлом должно возвращаться значение a^y (одно на всё сообщение), допускается в качестве значения y использовать предопределённый набор однобайтных величин. При расшифровании вместе с телом файла на микроконтроллер должна передаваться тройка (a, p, a^y) и микроконтроллер должен выполнять поиск секретного ключа по записям в EEPROM. Программа для МК должна осуществлять постоянное отображение открытых ключей криптосистемы, хранящихся в EEPROM, на семисегментном индикаторе. По прерыванию int0 должен циклически сменяться выводимый ключ, по прерыванию int1 должно производиться переключение между параметрами a и p. В качестве параметров для проверки работоспособности можно использовать следующие величины: $a = 6397$, $p = 8963$ и $x = 7499$.
6	Криптосистема RSA. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (n, e, d). При шифровании, расшифровании, подписании и проверке подписи открытый ключ (n, e) должен передаваться вместе с телом файла (хэшем) и при расшифровании и подписании микроконтроллер должен выполнять поиск секретного ключа по записям открытых ключей в EEPROM. Программа для МК должна отображать следующую статистику на семисегментном индикаторе: общую длину зашифрованных файлов, расшифрованных файлов и количество операций подписания и проверки подписи в виде E.xxx, d.xxx, S.xxx, C.xxx (e – encrypt, d – decrypt, s – sign, c – check sign). Переключение между выводимыми параметрами должно производиться циклически с помощью прерывания int0. В качестве параметров для проверки работоспособности можно использовать следующие величины: $n = 39913$, e = 453 и $d = 1221$.
7	Протокол Эль-Гамаля с открытым ключом. При шифровании необходимо использовать функцию $f(m, b^y) = m + b^y \pmod p$. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (a, p, x). При шифровании файла на контроллер должна передаваться пара (a, p), в ответном сообщении вместе с зашифрованным файлом должно возвращаться значение a^y (одно на всё сообщение), допускается в качестве значения y использовать

	предопределённый набор однобайтных величин. При расшифровании вместе с телом файла на микроконтроллер должна передаваться тройка (a, p, a^y) и микроконтроллер должен выполнять поиск секретного ключа по записям в EEPROM. Каждый набор ключей в EEPROM должен сопровождаться pin-кодом в виде последовательности из четырёх десятичных цифр. При операции расшифрования или подписания пользователь должен вводить pin-код на ПК и до выполнения операции введённый pin-код должен передаваться на МК, при несовпадении pin-кода операция не выполнятся, на ПК возвращается сообщение об ошибке и передача данных не выполняется. В качестве параметров для проверки работоспособности можно использовать следующие величины: $a = 6397$, $p = 8963$ и $x = 7499$.
8	Криптосистема RSA. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (n, e, d). При шифровании, расшифровании, подписании и проверке подписи открытый ключ (n, e) должен передаваться вместе с телом файла (хэшем) и при расшифровании и подписании микроконтроллер должен выполнять поиск секретного ключа по записям открытых ключей в EEPROM. Программа для МК должна вести подсчёт числа обращений с отсутствующим в памяти открытым ключом. При последовательном поступлении трёх таких обращений криптосистема должна блокироваться и на дальнейшие запросы шифрования/расшифрования выдавать соответствующее сообщение на ПК. Необходимо предусмотреть команду разблокировки криптосистемы в программе для ПК, при поступлении этой команды на МК пользователю необходимо ввести комбинацию кнопок на портах A-D (комбинация должна храниться в EEPROM, включать не менее трёх одновременно нажатых кнопок и двух задействованных портов), после ввода комбинации на ПК передаётся сигнал о разблокировке. В качестве параметров для проверки работоспособности можно использовать следующие величины: $n = 39913$, e = 453 и $d = 1221$.
9	Протокол Эль-Гамаля с открытым ключом. При шифровании необходимо использовать функцию $f(m, b^y) = m + b^y \pmod p$. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (a, p, x). При шифровании файла на контроллер должна передаваться пара (a, p), в ответном сообщении вместе с зашифрованным файлом должно возвращаться значение a^y (одно на всё сообщение), допускается в качестве значения y использовать предопределённый набор однобайтных величин. При расшифровании вместе с телом файла на микроконтроллер должна передаваться тройка

	(a, p, a^y) и микроконтроллер должен выполнять поиск секретного ключа по записям в EEPROM. МК должен вести подсчёт статистики по количеству команд шифрования и расшифрования по всем существующим в EEPROM ключам, а также единый набор по несуществующим ключам. Должна быть реализована возможность отображения на семисегментном индикаторе всей статистики по всем ключам: при помощи прерывания int0 происходит переключение между выводимым параметром статистики, с помощью прерывания int1 происходит переключение между выводимым ключом. Примеры вывода: 1.E.xx, 2.d.xx, -.E.xx, e – encrypt, d – decrypt, 1/2 – номер ключа, «-» – несуществующий ключ, xx – количество операций. В качестве параметров для проверки работоспособности можно использовать следующие величины: $a = 6397$, $p = 8963$ и $x = 7499$.
10	Криптосистема RSA. Приложение для микроконтроллера должно позволять одновременно хранить в памяти EEPROM несколько троек параметров (n, e, d). При шифровании, расшифровании, подписании и проверке подписи открытый ключ (n, e) должен передаваться вместе с телом файла (хэшем) и при расшифровании и подписании микроконтроллер должен выполнять поиск секретного ключа по записям открытых ключей в EEPROM. Приложение для ПК должно иметь возможность отправки на МК запроса, по которому будут переданы все открытые ключи, после чего вместо ввода параметров шифрования пользователь будет выбирать один из существующих открытых ключей. В качестве параметров для проверки работоспособности можно использовать следующие величины: $n = 39913$, e = 453 и $d = 1221$.