

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Новгородский государственный университет
имени Ярослава Мудрого

Гурьянов С.А.

Система отладки AVR микроконтроллеров ATmega16

Новгород

2012

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Новгородский государственный университет имени Ярослава Мудрого»
Институт электронных и информационных систем

Кафедра радиосистем

Гурьянов С.А.

Система отладки AVR микроконтроллеров ATmega16

Методическое пособие по курсу лабораторных работ
«Цифровые устройства и микропроцессоры»
«Информационно-измерительные и управляющие системы»

Великий Новгород

2012

УДК 621.38

ББК

Гурьянов С.А. Система отладки AVR микроконтроллеров ATmega16 :Методическое пособие /ФГБОУ «Новгородский государственный университет им. Ярослава Мудрого», Великий Новгород, 2012 г. - 53 с.

В учебном пособии рассмотрена структура лабораторного макета на основе микроконтроллера Atmega16, система команд микроконтроллера, примеры программ, приведено описание устройства ввода-вывода.

Учебное пособие отвечает новым образовательным стандартам и предназначено для подготовки специалистов по специальности ФГОС 210400.62 и ФГОС 210400.68.21 "Радиотехника" (Системы и устройства передачи, приема и обработки сигналов).

ВВЕДЕНИЕ	5
1. ОПИСАНИЕ ЛАБОРАТОРНОГО МАКЕТА.....	6
1.1 Структура микроконтроллера Atmega16	6
1.2 Назначение выводов микроконтроллера Atmega16.....	7
1.3 Описание таймеров/счетчиков	9
1.4 Схема лабораторного макета	15
1.5 Включение лабораторного макета	20
2. ПРОГРАММА ОТЛАДКИ AVR Studio.....	21
2.1 Запуск программы	21
2.2 Создание, редактирование, компиляция и отладка программы	23
3. ПРОГРАММА PonyProg2000	25
4. ОПРЕДЕЛЕНИЕ НОМЕРА ВИРТУАЛЬНОГО ПОРТА USB.....	27
5. ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ.....	30
6. ПРИМЕРЫ ПРОГРАММ.....	31
6.1 Программа приема и передачи цифрового сигнала	31
6.2 Программа формирования цифрового сигнала	33
с программным опросом окончания работы таймера.....	33
6.3 Программа формирования цифрового сигнала	37
с работой таймера по прерыванию.....	37
6.4 Программа приема и передачи данных по каналу RS232	41
с использованием ожидания готовности	41
6.5 Программа приема и передачи данных по каналу RS232	43
с использованием прерываний	43
7. ТЕСТОВАЯ ПРОГРАММА	46
СПИСОК ЛИТЕРАТУРЫ	53

ВВЕДЕНИЕ

Методическое пособие предназначено студентам специальности

- Радиотехника - ФГОС 210400.62 в курсе «Цифровые устройства и микропроцессоры»
- Радиотехника (Системы и устройства передачи, прием и обработки сигналов) - ФГОС 210400.68.21 в курсе «Информационно-измерительные и управляющие системы»

В пособии рассматривается лабораторный макет, реализованный с использованием 8-разрядного AVR микроконтроллера фирмы ATMEL Atmega16.

Приводится описание данного микроконтроллера и основных блоков.

Представлено описание схемы электрической принципиальной лабораторного макета и его конструкции.

Приводится описание необходимого программного обеспечения, используемого для написания программ и их компиляции (AVR Studio), а также для непосредственного программирования микроконтроллера (Pony Prog).

Представлены примеры программ для работы с цифровыми сигналами, внутренними таймерами и интерфейсом RS232. Приводится тестовая программа для проверки работоспособности всех основных блоков лабораторного макета.

1. ОПИСАНИЕ ЛАБОРАТОРНОГО МАКЕТА

Лабораторный макет предназначен для изучения системы команд и методов программирования AVR микроконтроллеров фирмы Atmel. Лабораторный макет реализован на базе микроконтроллера Atmega16.

1.1 Структура микроконтроллера Atmega16

На рис. 1 изображена внутренняя структура микроконтроллера Atmega16.

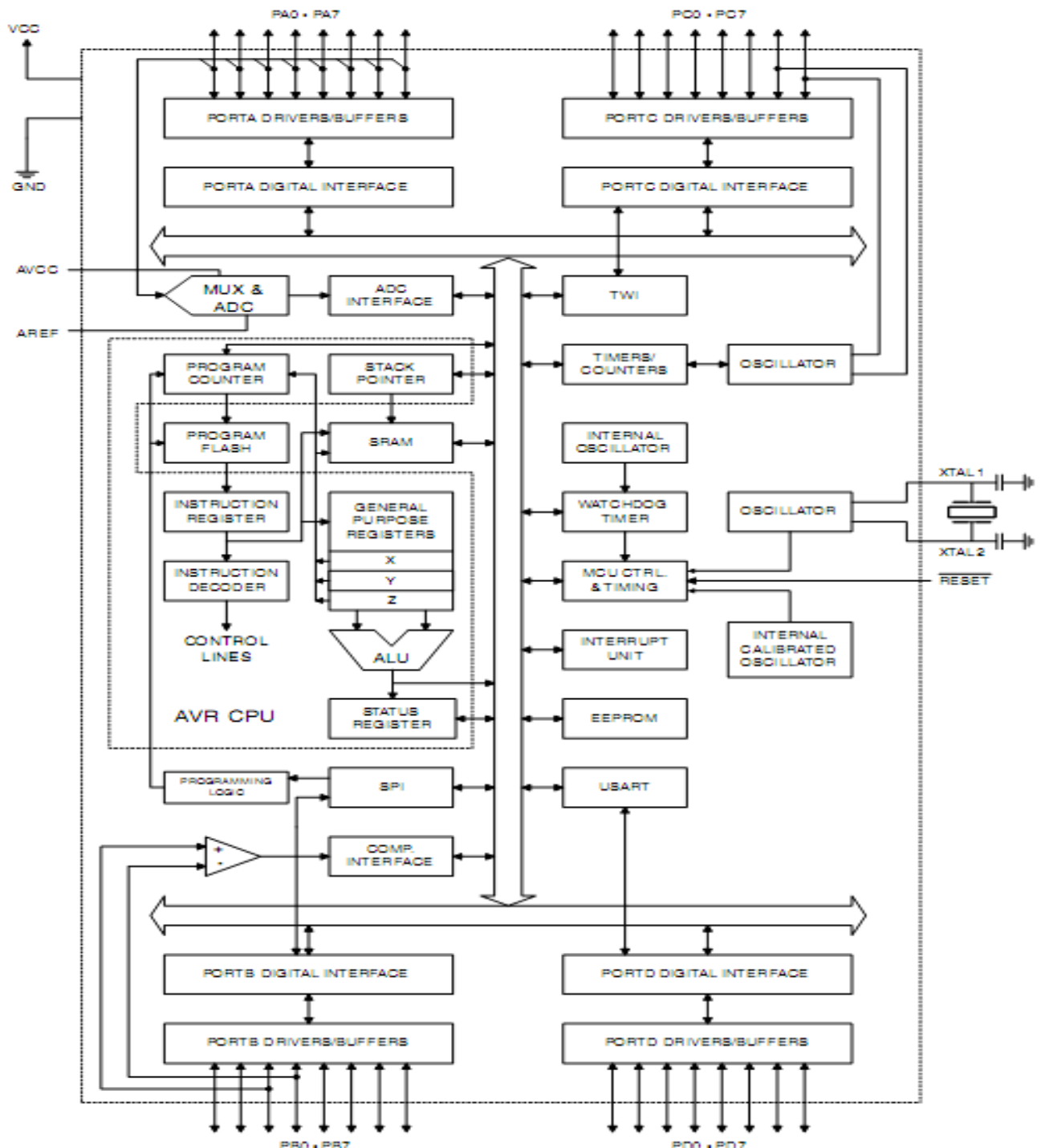


Рис.1. Внутренняя структура микроконтроллера Atmega16

1.2 Назначение выводов микроконтроллера Atmega16

Микроконтроллер Atmega16 имеет 4 порта ввода-вывода – port A (PA7 – PA0), port B (PB7 – PB0), port C (PC7 – PC0) и port D (PD7 – PD0). Порты являются двунаправленными портами ввода-вывода с внутренними подтягивающими резисторами, устанавливаемыми для каждого бита в отдельности.

Порты ввода-вывода также могут выполнять специальные функции для различных устройств ввода-вывода:

- Port A (PA7 – PA0) может использоваться как восьми разрядный аналоговый вход для АЦП. В этом случае

AVCC - является контактом питания АЦП.

Если АЦП не используется, то этот вывод должен быть соединен с источником питания напрямую, а в случае использования АЦП, подключение к источнику питания осуществляется через НЧ фильтр.

AREF – опорное напряжение АЦП.

- Port B может использовать
PB0(T0) и PB1(T1) - два входа таймеров,
PB2(AIN0) и PB3(AIN1) - два входа аналоговых компараторов,
PB4(SS), PB5(MOSI), PB6(MISO), PB7(SCK) – выводы для подключения интерфейса SPI.
- Port C может использовать
PC0(SCL), PC1(SDA) – выводы интерфейса I2C,
PC5(TDI), PC4(TDO), PC3(TMS), PC2(TCK) – выводы для подключения интерфейса JTAG.
- Port D может использовать
PD0(RXD), PD1(TXD) - выводы интерфейса UART (RS232),
PD2(INT0), PD3(INT1) - два входа внешних прерываний.

Специальные выводы микроконтроллера Atmega16:

RESET – вывод сброса с активным низким уровнем напряжения,

XTAL1 – вход для подключения кварцевого резонатора,

XTAL2 – выход для подключения кварцевого резонатора,

VCC и GND – подключение источника питания и общего провода.

На рис.2 представлено расположение контактов микроконтроллера Atmega16 для двух различных вариантов корпусов (TQFP и PDIP). В лабораторном макете применена микросхема в корпусе PDIP.

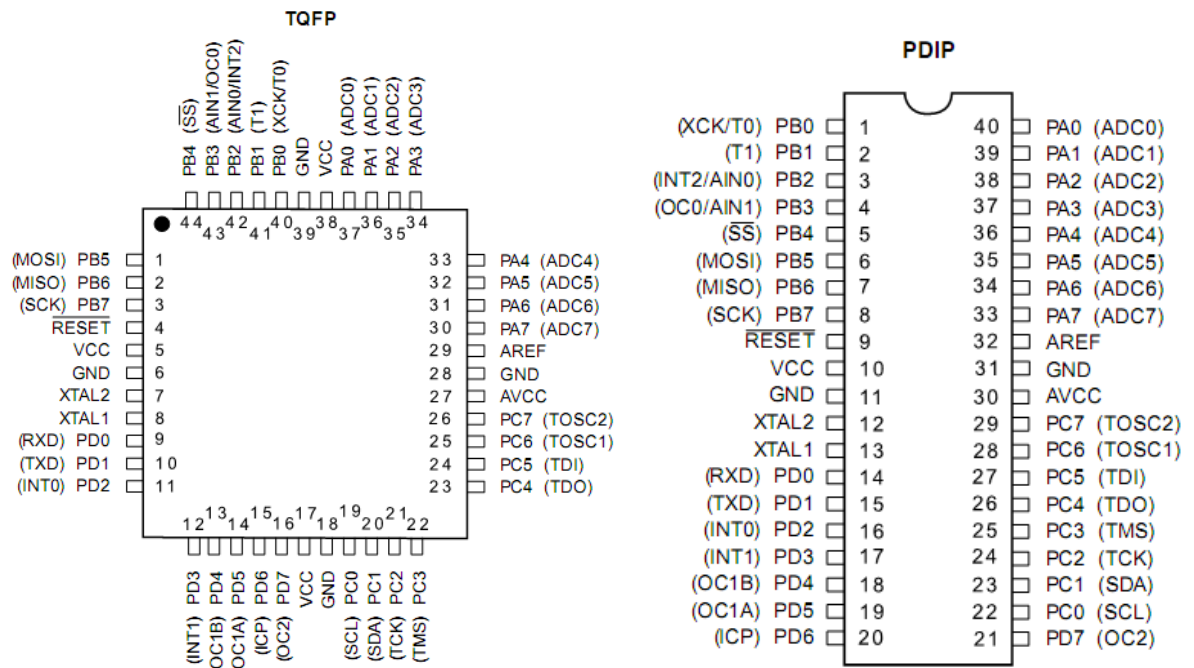


Рис. 2. Расположение контактов микроконтроллера Atmega16 для корпусов типа TQFP и PDIP.

1.3 Описание таймеров/счетчиков

Микроконтроллеры ATmega16 оснащены тремя таймерами/счетчиками общего назначения - двумя 8-разрядными T/C0 и T/C2 и одним 16-разрядным T/C1.

Таймеры-счетчики 0 и 2 - модули многофункционального одноканального 8-разрядного таймера-счетчика с аппаратным выходом для генерации ШИМ-сигнала и встроенным асинхронным опциональным тактовым генератором, который оптимизирован под использование часового кварца (32768Гц) для асинхронного по отношению к системной синхронизации тактирования.

Основные отличительные особенности таймера-счетчика 0 и 2:

- Одноканальный счетчик
- Опциональный режим сброса таймера при совпадении (автоматическая перезагрузка)
- Широтно-импульсная модуляция без генерации ложных импульсов при записи нового порога сравнения в OCR0 (двойная буферизация) и с фазовой коррекцией
- Генератор частоты
- 10-разрядный предделитель тактовой частоты
- Генерация прерываний по переполнению и выполнения условия сравнения (TOV0 и OCF0)
- Возможность асинхронного тактирования совместно с внешним кварцевым резонатором частотой 32 кГц независимо от частоты синхронизации ввода-вывода

16-разрядный таймер-счетчик 1 предназначен для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов.

Таймер/счетчик 1 может работать

- в обычном режиме прямого счета до переполнения,
- в двух режимах сравнения выхода, используя регистры сравнения выходов А или В - OCR1A и OCR1B в качестве источников данных, сравниваемых с содержимым таймера/счетчика 1,
- в режиме захвата содержимого таймера/счетчика 1 в регистр захвата входа.

В качестве входного сигнала таймера/счетчика 1 может использоваться

- системный тактовый сигнал СК,
- масштабированный системный тактовый сигнал СК после делителя с заданным коэффициентом деления,
- внешний тактовый сигнал, поступающий на вход T1 микроконтроллера.

Таймер/счетчик 1 формирует четыре прерывания

- при переполнении,
- при совпадении значений счетчика и регистров сравнения А или В,
- при захвате событий.

Блок-схема таймера/счетчика 1 приведена на рис. 3.

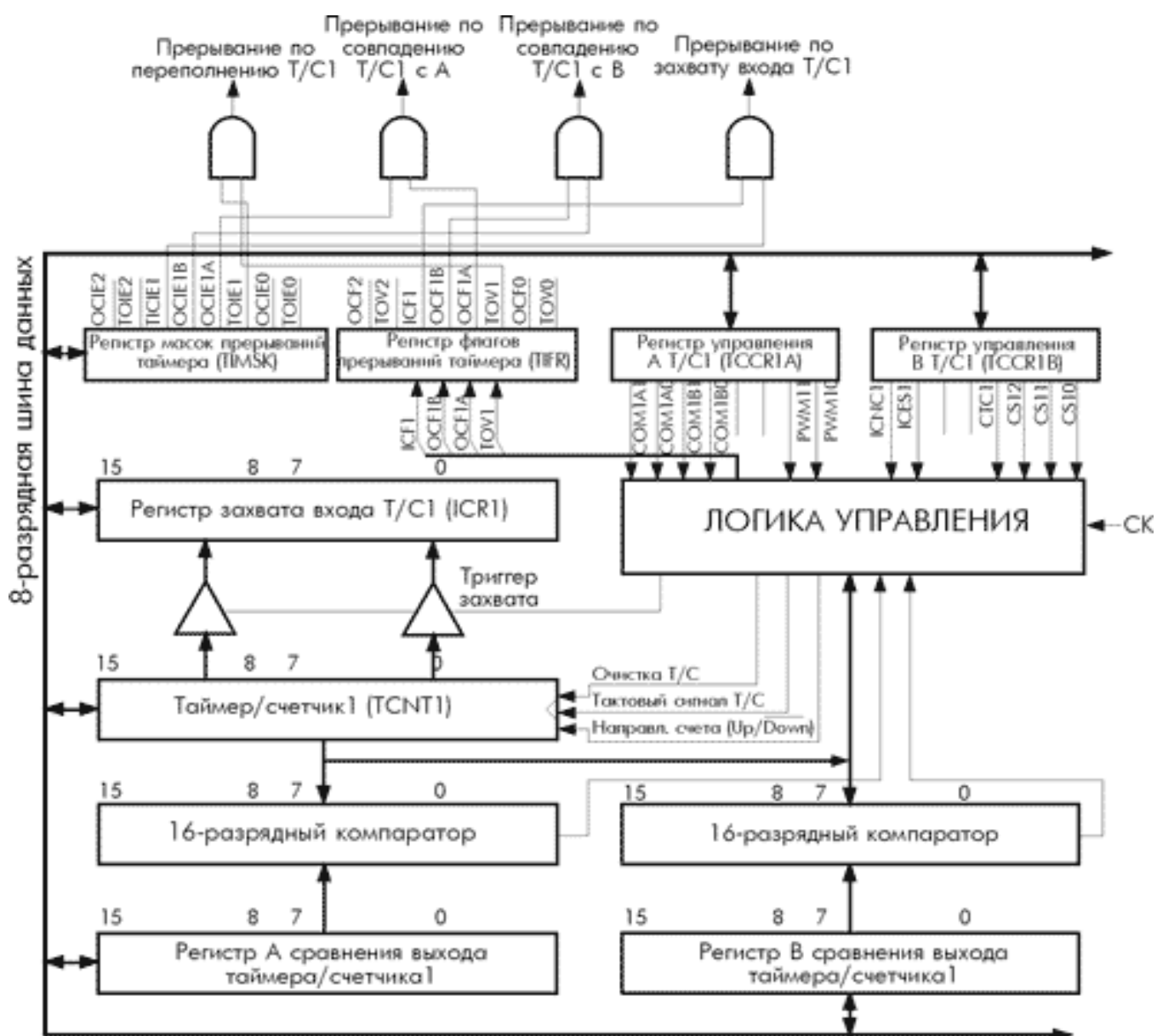


Рис.3. Блок-схема таймера/счетчика 1

Управление работой таймера/счетчика 1 осуществляется с помощью регистров, описанных в таблице 1.

Таблица 1 – Описание регистров управления таймера/счетчика 1

Регистр	Краткое описание
TCNT1	Timer/Counter1 - регистр, содержащий текущее значение таймера счетчика 1.
TCCR1A	Timer/Counter1 Control Register A – регистр задания режимов таймера/счетчика 1
TCCR1B	Timer/Counter1 Control Register B – регистр задания режимов таймера/счетчика 1
OCR1A	Timer/Counter Output1 Compare Register A - выходной регистр компаратора A
OCR1B	Timer/Counter Output1 Compare Register B - выходной регистр компаратора B
ICR1	Timer/Counter1 Input Capture Register1 - входной регистр защелки 1-го таймера
TIMSK	Timer Interrupt Mask Register - регистр маски прерываний
TIFR	Timer Interrupt Flag Register - регистр флагов прерываний

16-битный регистры TCNT1 содержит текущее значение таймера счетчика 1. Составляет из старшего TCNT1H (Timer counter 1 High byte) и младшего регистра TCNT1L (Low byte). Для копирования регистра TCNT1 при побайтной работе используется регистр ICR1.

Регистры задания режимов таймера/счетчика 1 TCCR1A и TCCR1B определяют режимы работы таймера/счетчика 1.

Регистр задания режимов таймера/счетчика 1 TCCR1A имеет следующие биты управления:

7	6	5	4	3	2	1	0
COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Биты COM1A1, COM1A0, COM1B1 и COM1B0 изменяют режим формирования выходного сигнала OC1A и OC1B.

Биты FOC1A, FOC1B принудительной установки результатов сравнения каналов А и В.
Биты WGM11 и WGM10 служат для задания режима работы таймера/счетчика 1.

Регистр задания режимов таймера/счетчика 1 TCCR1B имеет следующие биты управления:

7	6	5	4	3	2	1	0
CICNC1	ICSE1	-	WGM13	WGM12	CS12	CS11	CS10

Бит ICNC1 активизирует подавитель шума на входе захвата.

Бит ICES1 позволяет задать, какой фронт на входе захвата ICP1 приведет к захвату со-

Биты WGM13 и WGM12 служат для задания режима работы таймера/счетчика 1.

стояния таймера (совместно с WM11 и WGM10).

Биты CS12, CS11 и CS10 программируют выбор источника сигнала, коэффициент деления предварительного делителя, а также осуществляют запуск и остановку счетчика в соответствии с таблицей 2.

Таблица 2 – Коэффициенты предварительного делителя

CS12	CS11	CS10	Краткое описание
0	0	0	Счет остановлен
0	0	1	CLK
0	1	0	CLK/8
0	1	1	CLK/64
1	0	0	CLK/256
1	0	1	CLK/1024
1	1	0	Работа с внешним сигналом T1 – работа по переднему фронту
1	1	1	Работа с внешним сигналом T1 – работа по заднему фронту

При задании необходимого значения коэффициента деления или задании режима работы с внешним сигналом происходит **включение** таймера/счетчика 1.

При задании коэффициентов CS12, CS11 и CS10 равным 0 происходит **выключение** таймера/счетчика 1.

В таблице 3 представлены режимы работы таймера/счетчика1.

Таблица 3 – режимы работы таймера/счетчика1

Номер режима	WGM13	WGM12 (CTC12)	WGM11 (pwm11)	WGM10 (pwm10)	Режим работы	Верхний предел счета	Обновление OCR1x	Установка Флага TOV1 на:
0	0	0	0	0	Нормальный	0xFFFF	сразу после записи	МАКС
1	0	0	0	1	8-разр. ШИМ ФК	0x00FF	на вершине счета	нижнем пределе
2	0	0	1	0	9-разр. ШИМ ФК	0x01FF	на вершине счета	нижнем пределе
3	0	0	1	1	10-разр. ШИМ ФК	0x03FF	на вершине счета	нижнем пределе
4	0	1	0	0	СТС	OCRnA	сразу после записи	МАКС
5	0	1	0	1	8-разр. быстрая ШИМ	0x00FF	на вершине счета	на вершине счета
6	0	1	1	0	9-разр. быстрая ШИМ	0x01FF	на вершине счета	на вершине счета
7	0	1	1	1	10-разр. быстрая ШИМ	0x03FF	на вершине счета	на вершине счета
8	1	0	0	0	ШИМ ФЧК	ICRn	на нижнем пределе	нижнем пределе
9	1	0	0	1	ШИМ ФЧК	OCRnA	на нижнем пределе	нижнем пределе
10	1	0	1	0	ШИМ ФК	ICRn	на вершине счета	нижнем пределе
11	1	0	1	1	ШИМ ФК	OCRnA	на вершине счета	нижнем пределе
12	1	1	0	0	СТС	ICRn	сразу после записи	МАКС.
13	1	1	0	1	(резерв)	–	-	-
14	1	1	1	0	Быстрая ШИМ	ICRn	на вершине счета	на вершине счета
15	1	1	1	1	Быстрая ШИМ	OCRnA	на вершине счета	на вершине счета

16-битные регистры OCR1A и OCR1B служат для задания значения, при достижении которого в режиме счёта происходит их сравнение, и таймер/счетчик генерирует соответствующие прерывания по совпадению.

Регистр флагов прерывания таймеров/счетчиков TIFR устанавливает биты прерывания всех трех (0,1,2) таймеров микроконтроллера и имеет следующие биты:

7	6	5	4	3	2	1	0
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

Бит 2- TOV1 -При переполнении регистра TCNT1 таймер/счетчик 1 устанавливает флаг TOV1 (Overflow) – переполнение таймера/счетчика 1.

Бит 3- OCF1B -При совпадении значений в регистрах TCNT1 и OCR1B формируются флаг OCF1B (Output Compare B Match).

Бит 4- OCF1A - При совпадении значений в регистрах TCNT1 и OCR1A формируются флаг OCF1A (Output Compare A Match).

Бит 5- ICF1 - При захвате входного значения формируется флаг ICF1 (Input Capture).

Вызов соответствующих прерываний зависит от значения соответствующих битов регистра TIMSK и флага I в Status регистре процессора.

Регистр масок прерываний TIMSK имеет следующие бит, которые по расположению совпадают с битами регистра флагов прерываний TIFR:

7	6	5	4	3	2	1	0
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

Бит 2 - TOIE1 - Разрешение прерывания на сигнал переполнения таймера/счетчика 1, вызывается прерывание по вектору \$010 (OVF1addr).

Бит 3 - OCIE1B - Разрешение прерывания при совпадении счёта с константой в регистре OCR1B, вызывается прерывание по вектору \$00E (OC1Baddr).

Бит 4 - OCIE1A - Разрешение прерывания при совпадении счёта с константой в регистре OCR1A, вызывается прерывание по вектору \$00C (OC1Aaddr).

Бит 5 - TICIE1 - Разрешение прерывания захвата таймера/счетчика 1, расположенного по вектору \$00A (ICP1addr).

1.4 Схема лабораторного макета

На рис. 4 представлена верхняя панель лабораторного макета, а на рис. 5 (стр. 16 и стр. 17) - схема электрическая принципиальная лабораторного макета.

Схему лабораторного макета можно разделить на следующие основные модули:

- Модуль микроконтроллера,
- Модуль программирования,
- Модуль обработки цифровых сигналов,
- Модуль обработки аналоговых сигналов,
- Модуль последовательного интерфейса,
- Модуль USB (виртуальный последовательный интерфейс),
- Модуль клавиатуры,
- Модуль питания.

Модуль микроконтроллера является основным модулем лабораторного макета и реализован на микросхеме D3 Atmega16 по стандартной схеме включения. Частота кварца лабораторного макета равна 11.0592 МГц.

Программирование микроконтроллера осуществляется через разъем специализированного последовательного интерфейса (разъем X1), расположенного на задней панели макета. Программирование осуществляется по стандартной схеме с помощью программы PonyProg.

Модуль обработки цифровых сигналов реализован на базе двунаправленного буфера D1. Входные цифровые сигналы подаются на «BX1» и «BX2» лабораторного макета, которые на схеме обозначены как X8(IN1) и X7(IN2). Выходные цифровые сигналы микроконтроллера снимаются с соответствующих выходов «ВЫХ 1» и «ВЫХ 2», обозначенные на схеме как X3(OUT2) и X4(OUT1). Все эти сигналы поступают на микроконтроллер через буферный регистр. Для работы с использованием встроенного таймера микроконтроллера используется вход таймера «BX_T» «BX3» и выход таймера «ВЫХ_T» «ВЫХ3», буферизированные с помощью микросхемы D1. Далее описаны используемые разряды порта D и порта В микроконтроллера:

```
.equ RXD = 0           // PD.0 вход последовательного интерфейса
.equ TXD = 1           // PD.1 выход последовательного интерфейса

.equ in1 = 2           // PD.2 вход 1 «BX1»
.equ in2 = 3           // PD.3 вход 2 «BX2»
.equ out1 = 4          // PD.4 выход 1 «ВЫХ1»
.equ out2 = 5          // PD.5 выход 2 «ВЫХ2»
```

```
.equ WR = 6           // PD.6 сигнал записи в ЦАП
.equ SEL = 7          // PD.7 сигнал выбора ЦАПА A/B

.equ in_tmr1 = 1       // PB.1 вход таймера «ВХЗ»
.equ out_tmr1 = 3      // PB.3 выход таймера «ВЫХЗ»
```

Следует отметить, что выходы буферного регистра O1, O2 и O_T1 жестко соединены с выводами микроконтроллера (номер 16 – PD4, 17 – PD5 и 2 – PB3). Поэтому при программировании микроконтроллера эти выводы запрещается программировать как выходные. Для программирования направления работы данного порта необходимо использовать следующую конфигурацию

Idi tmp,0b11110010 .

Для работы с аналоговыми сигналами предусмотрен модуль обработки аналоговых сигналов, реализованный с использованием встроенного в микроконтроллер АЦП и внешнего ЦАП с буферными усилителями.

При работе с АЦП входной сигнал подается на разъем «АЦП» X9, и через два усилительных каскада, реализованных на операционном усилителе D6, поступает на вход ADC0 АЦП микроконтроллера. Входные операционные усилители осуществляют буферизацию входного сигнала и смещение нулевого уровня входного сигнала на уровень половины опорного напряжения АЦП. **Размах входного напряжения не должен превышать 5В.**

Выходным сигналом аналогового модуля являются два выхода ЦАП «ЦАП 1» и «ЦАП 2» (X10 и X11 соответственно). ЦАП реализован на двухканальной микросхеме D4. Выбор каналов ЦАП осуществляется сигналом SEL, а запись – сигналом отрицательной полярности WR, поступающими от микроконтроллера (PD.6). Данные на ЦАП поступают соответственно с порта C микроконтроллера. На операционных усилителях D7 реализованы буферные усилители ЦАП. Опорное напряжение одного из этих каскадов имеет положительное значение +5В, а второго – отрицательное значение -5В. Поэтому имеется возможность формирования положительных и отрицательных сигналов (относительно 0В).

Модуль последовательного интерфейса реализован на микросхеме D5, выполняющей функцию драйвера интерфейса RS232 и дает возможность подключения макета к персональному компьютеру.

Модуль USB на микросхеме D8 позволяет реализовать виртуальный последовательный интерфейс через разъем X2 (USB), выполняющий те же функции. Номер последовательного интерфейса определяется через функцию «Свойства» программы «Мой компьютер» (глава 4).

Модуль клавиатуры реализован в виде 4 кнопок «КН1»-«КН4», работающих на замыкание. Кнопки подключены к выводам порта А - PA4-PA7 соответственно.

Питание лабораторного макета осуществляется через разъем X2 (USB), расположенный на задней панели макета и подключенный к персональному компьютеру. Для обеспечения двух полярного напряжения питания, необходимого для корректной работы операционных усилителей, используется микросхема D2, являющаяся DC/DC преобразователем +5В в -5В.

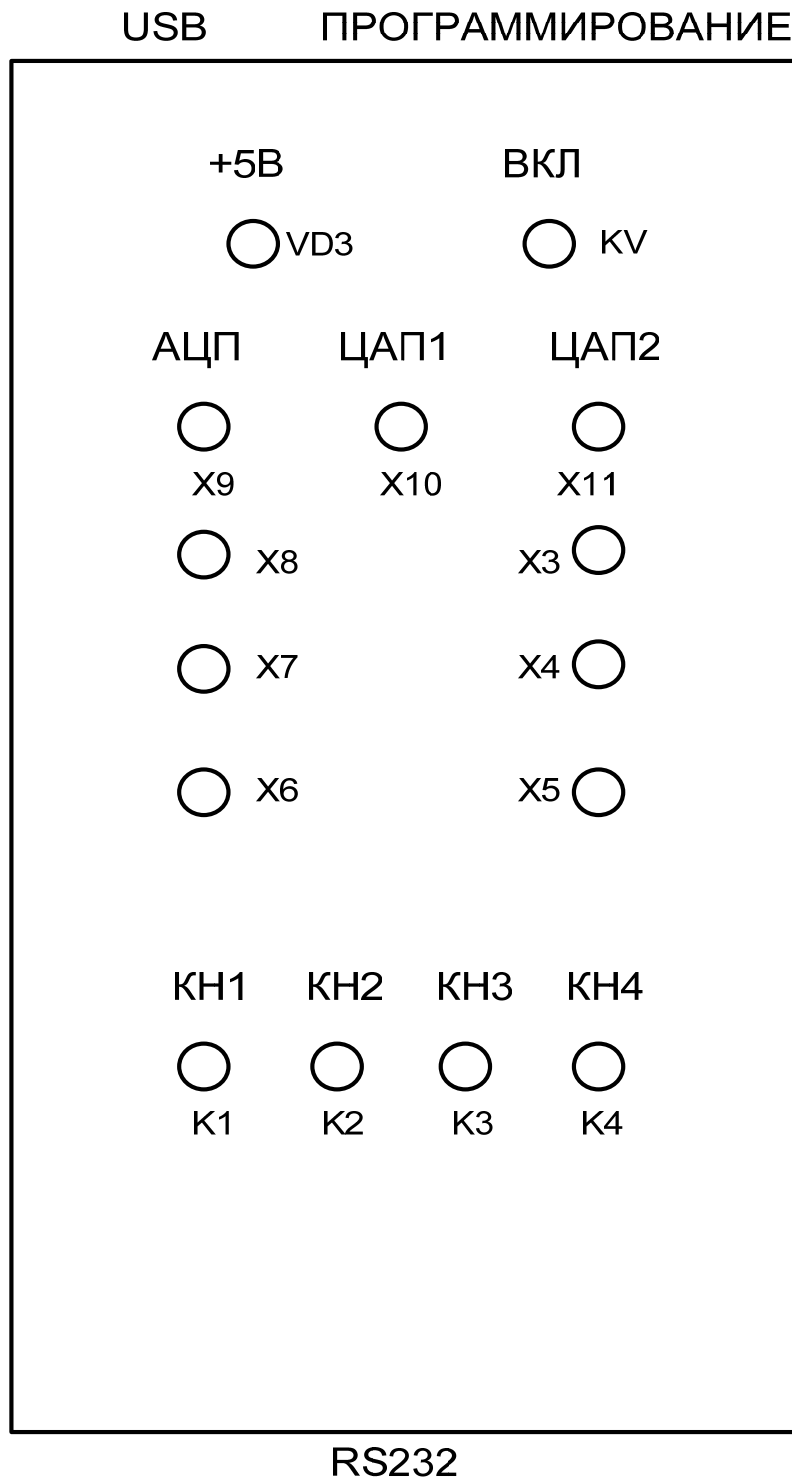
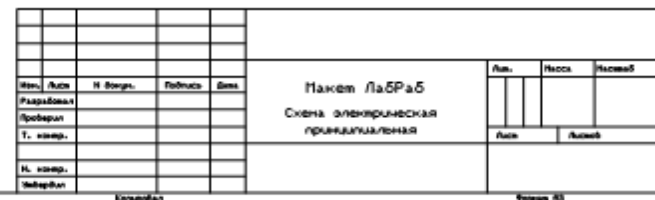
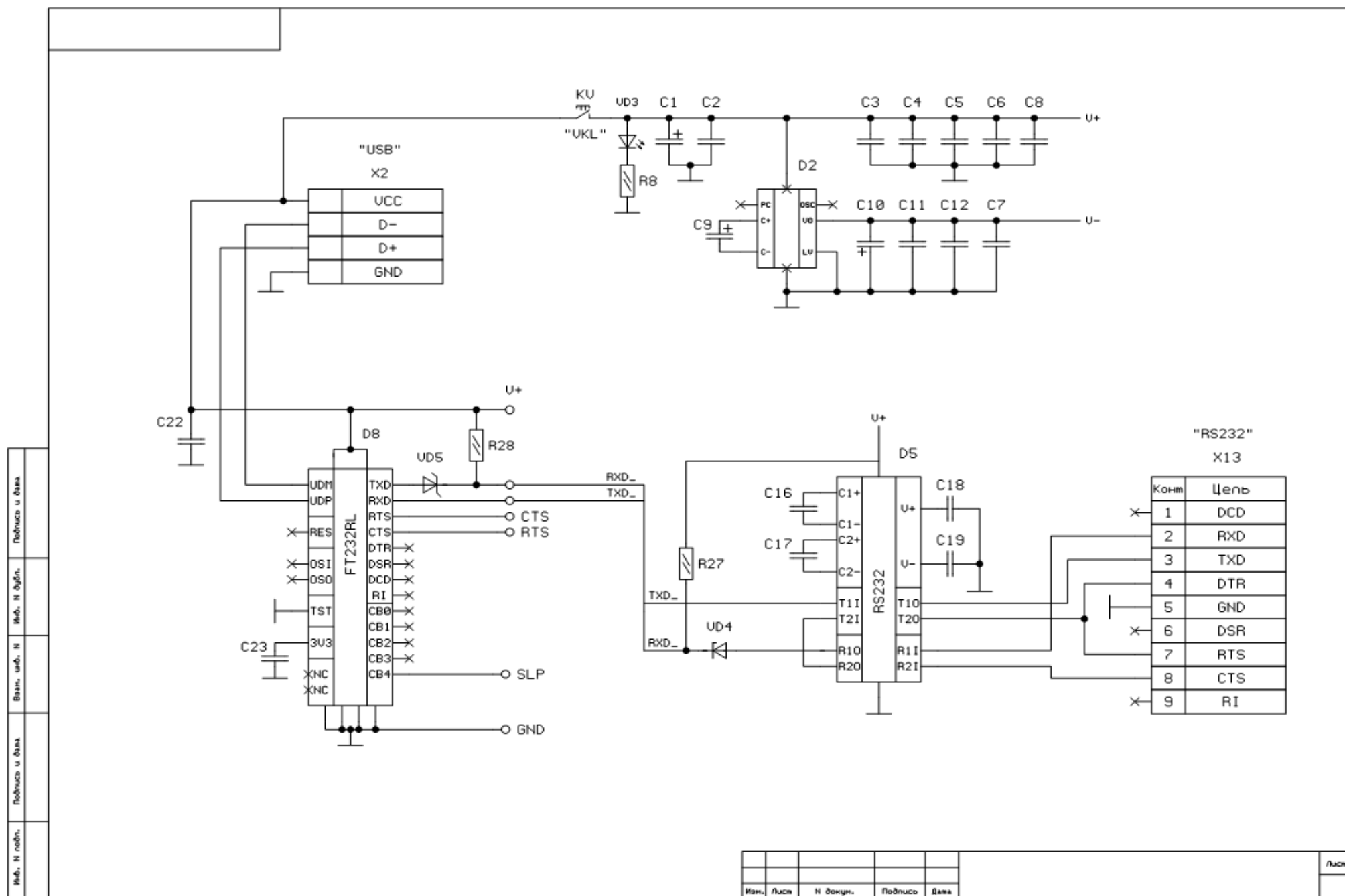


Рис.4. Верхняя панель лабораторного макета





1.5 Включение лабораторного макета

Для включения лабораторного макета необходимо соединить стандартным кабелем USB разъем макета с надписью «USB» с соответствующим разъемом персонального компьютера.

Для программирования контроллера необходимо соединить кабелем RS232 разъем макета с надписью «ПРОГРАММИРОВАНИЕ» с соответствующим разъемом персонального компьютера.

Включение макета осуществляется тумблером «ВКЛ». При этом загорается светодиод «+5В».

2. ПРОГРАММА ОТЛАДКИ AVR Studio

Программа AVR Studio предназначена для создания, редактирования, компиляции и отладки программного обеспечения для микроконтроллеров AVR.

2.1 Запуск программы

Для запуска программы следует выбрать пункт меню

«Пуск»-> «Программы» -> «Atmel AVR Tools» -> «AVR Studio 4»

После запуска данного приложения автоматически запустится мастер создания нового проекта. Если этого не произошло нужно в меню «Project» выбрать пункт «New Project» (см. рис. 2.1)

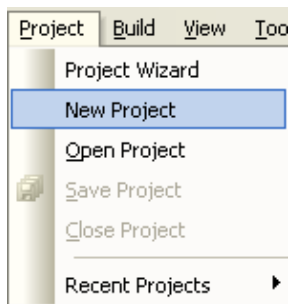


Рис. 2.1 – Меню “Project”

В мастере создания нового проекта требуется нажать кнопку «New Project» (см. рис. 2.2).

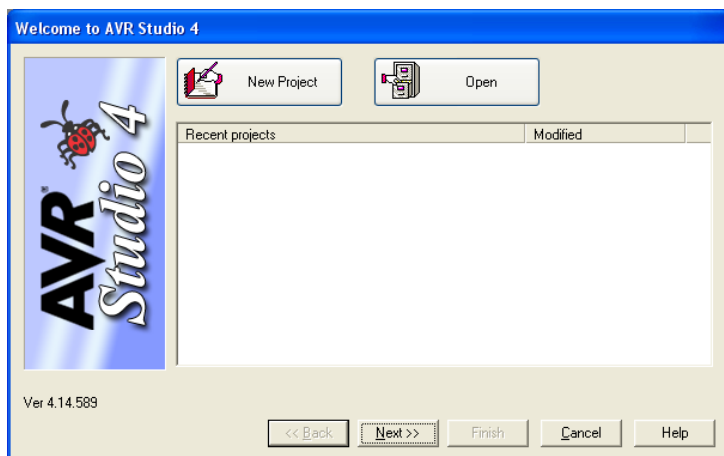


Рис. 2.2 – Мастер создания нового проекта (Шаг 1)

На следующем шаге нужно выбрать:

Язык программирования:

«Atmel AVR Assembler»

«AVR GCC» (для работы требуется WinAVR gcc plug-in).

выбрать папку для сохранения (путь к проекту не должен содержать пробелов и/или русских букв!).

Указать имя проекта (нужно проследить за тем, чтобы была установлена галочка возле «Create initial file» в этом случае имя файла с исходным текстом программы будет совпадать с именем проекта).

Пример установки настроек проекта приведен на рис. 2.3.

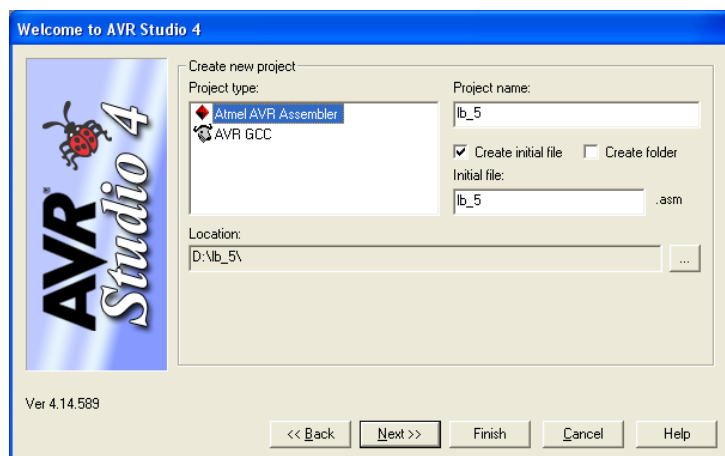


Рис. 2.3 – Мастер создания нового проекта (Шаг 2)

Далее необходимо выбрать платформу для отладки программы (Debug platform) и устройство (Device). В нашем случае это AVR Simulator и ATmega16 соответственно (см. рис. 2.4).

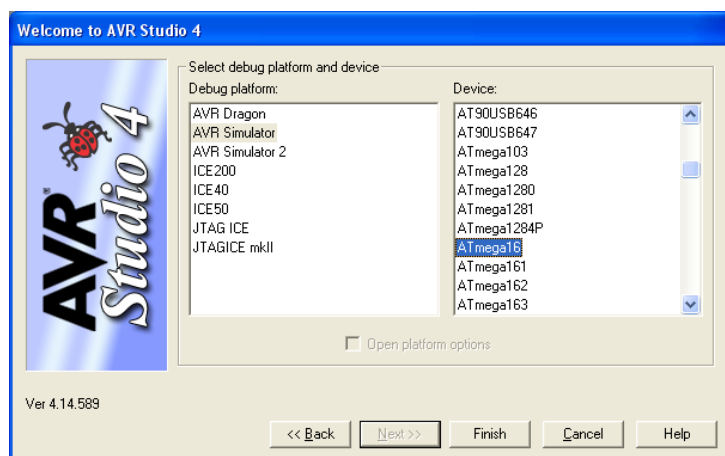


Рис. 2.4 – Мастер создания нового проекта (Шаг 3)

2.2 Создание, редактирование, компиляция и отладка программы

Рабочая область программы состоит из следующих частей (рис. 2.5):

- 1 - панель управления проектом
- 2 - окно ввода и правки текста программы
- 3 - окно вывода предупреждений и сообщений об ошибках
- 4 - панель для работы с устройствами ввода/вывода

Вывод дополнительных панелей осуществляется через меню «View» (см. рис.

2.6)

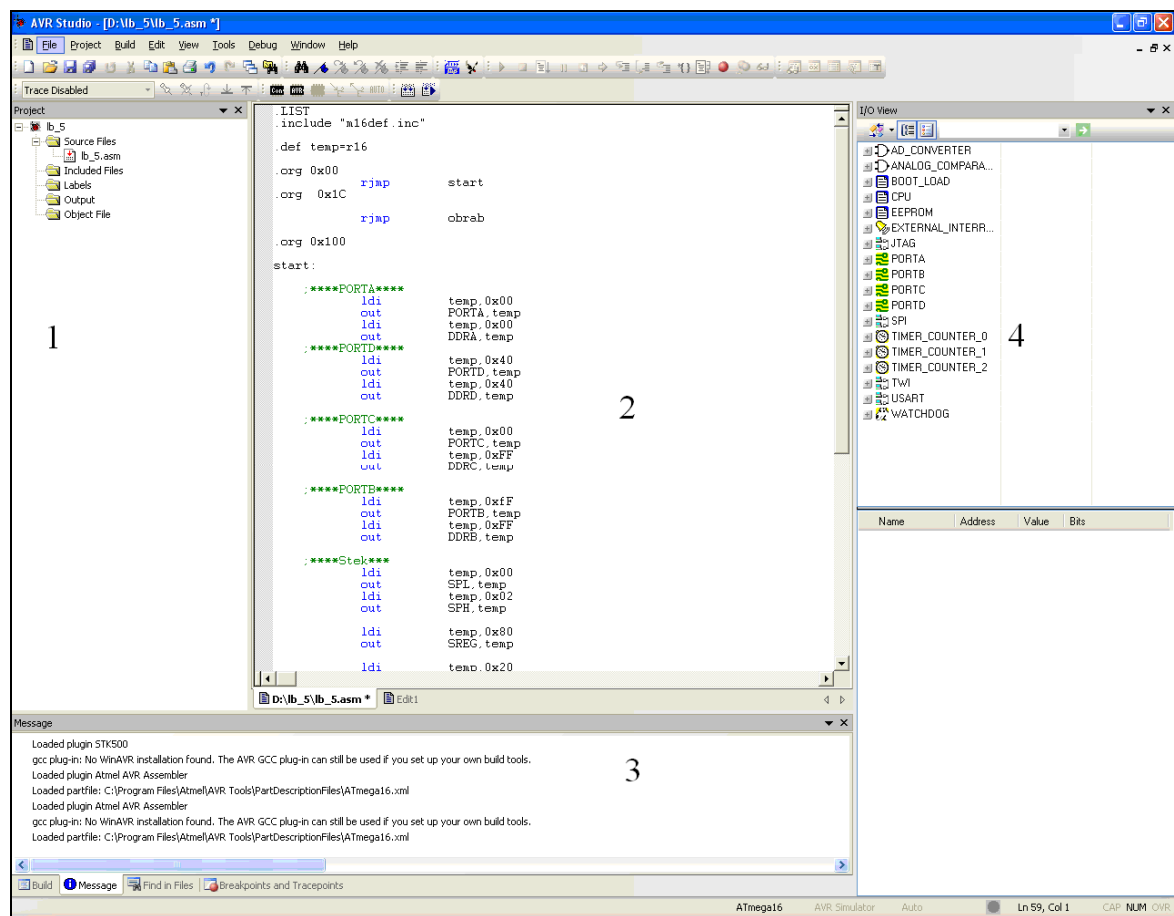


Рис. 2.5 – Рабочая область программы

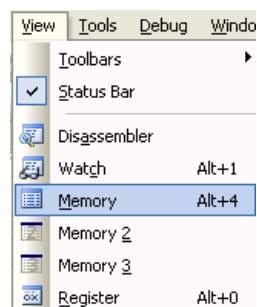


Рис. 2.6 – Меню «View»

Компиляция проекта осуществляется через меню «Build» (рис. 2.7). Чтобы откомпилировать проект нужно выбрать пункт «Build» или нажать клавишу «F7», для компиляции с последующей отладкой – «Build and Run» или использовать комбинацию клавиш «ALT + F7»

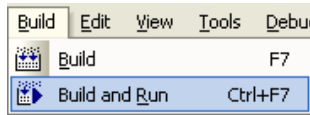


Рис. 2.7 – Меню «Build»

Для выполнения отладки служит меню «Debug» (см. рис. 2.8).

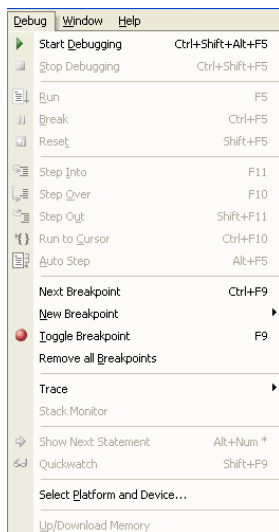


Рис. 2.8 - меню «Debug»

Наиболее часто используемыми являются пункты меню, описанны в таблице 1

Таблица 1 – Описание некоторых пунктов меню «Debug»

Пункт меню	Комбинация клавиш для быстрого запуска	Краткое описание
Run	F5	Запуск программы на выполнение
Break	Ctrl + F5	Остановка выполнения программы
Step Into	F11	Пошаговая трассировка с заходом в подпрограммы
Step Over	F10	Пошаговая трассировка без захода в подпрограммы
Toggle Breakpoint	F9	Установка точки останова ¹⁾
Select Platform and Device	-	Выбор типа платформы и устройства для отладки программы, если этого не было сделано при создании проекта

¹ Для этого нужно выделить требуемую строку с помощью мыши, а затем выбрать соответствующий пункт в меню.

3. ПРОГРАММА PonyProg2000

Программа PonyProg2000 предназначена для записи откомпилированной программы во Flash-память микроконтроллера. Также имеется возможность проверки записи и чтение содержимого памяти микроконтроллера.

Для запуска программы следует выбрать пункт меню «Пуск»-> «Программы» -> «PonyProg»
-> «PonyProg2000». Затем в меню «File» необходимо выбрать пункт «Open Device File» (см. рис. 3.1).

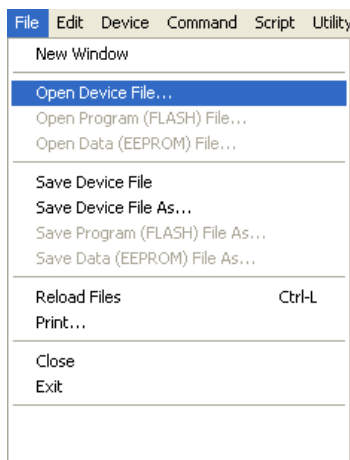


Рис. 3.1 – Меню «File»

В диалоговом окне «Open device content file» в поле «Имя файла» указать *.hex (см. рис. 3.2)

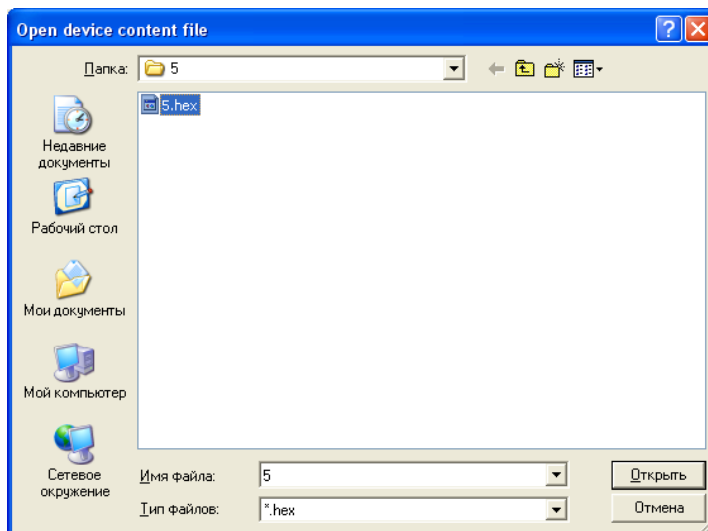


Рис. 3.2 – Окно «Open device content file»

После выполнения описанных выше действий основное окно программы будет выглядеть вот так (см. рис. 3.3)

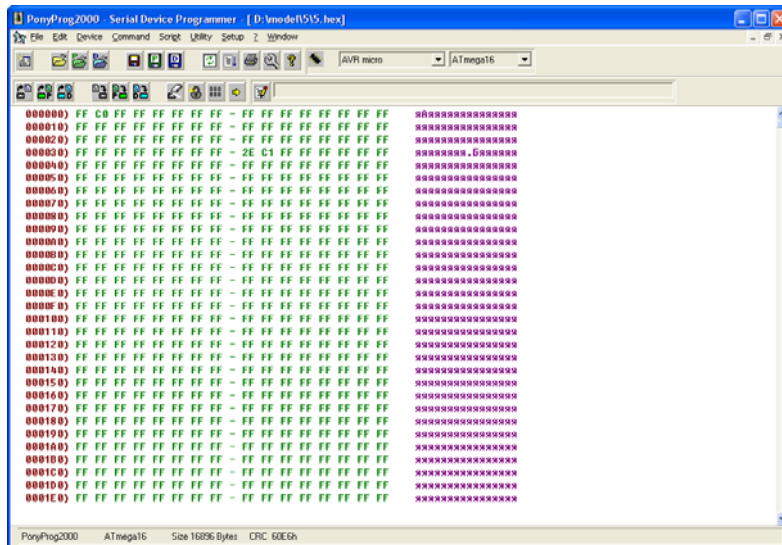


Рис. 3.3 – Основное окно программы

Чтобы указать программе порт, к которому будет подключен макет нужно в меню «Setup» выбрать пункт «Interface Setup» (рис. 3.4), затем установить переключатель на «COM1» (рис. 3.5).

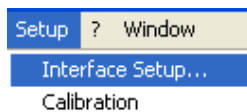


Рис. 3.4 – меню «Setup»

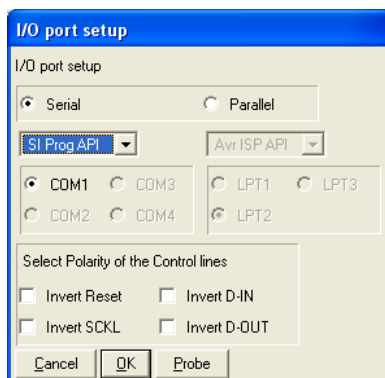


Рис. 3.5 – окно настроек

Заключительным этапом настройки будет выбор семейства и модели микроконтроллера на панели инструментов главного окна программы (см. рис. 3.6).



Рис. 3.6 – Выбор семейства и модели МК

Для начала загрузки программы в память МК нужно нажать кнопку .

4. ОПРЕДЕЛЕНИЕ НОМЕРА ВИРТУАЛЬНОГО ПОРТА USB

При отладке программ, использующих интерфейс RS232 лабораторного макета, можно использовать два порта последовательного интерфейса:

- Разъем X13 – разъем последовательного интерфейса RS232,
- Разъем X2 – разъем интерфейса USB (виртуальный последовательный интерфейс).

Одновременное использование двух последовательных портов невозможно.

При использовании разъема X13 макет подключается к соответствующему разъему последовательного интерфейса компьютера (номер 0 или 1). При наличии одного последовательного порта компьютера необходимо отключить кабель программирования макета и подключить кабель RS232 к соответствующему разъему.

При использовании разъема X2 (виртуального порта) макет уже подключен к разъему USB компьютера и необходимо только определить номер используемого последовательного порта с помощью функции «Диспетчер оборудования» персонального компьютера.

Для этого необходимо запустить правой клавишей мыши программу «Мой компьютер»-> «Свойства» (рис. 4.1).

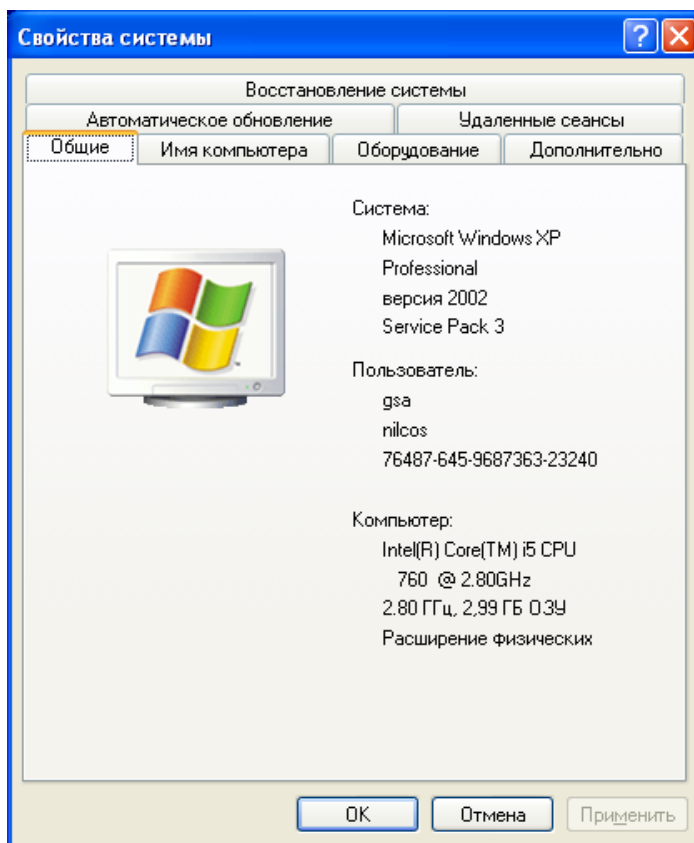


Рис. 4.1 – Окно «Свойства системы»

Выбрать пункт меню «Оборудование», и включить в нем функцию «Диспетчер устройств» (рис. 4.2).

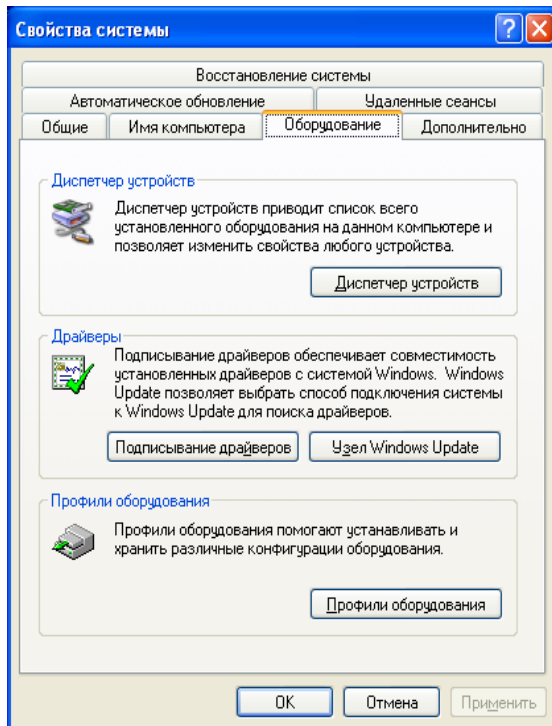


Рис. 4.2 – Окно «Оборудование»

Появится окно диспетчера устройств (рис. 4.3)

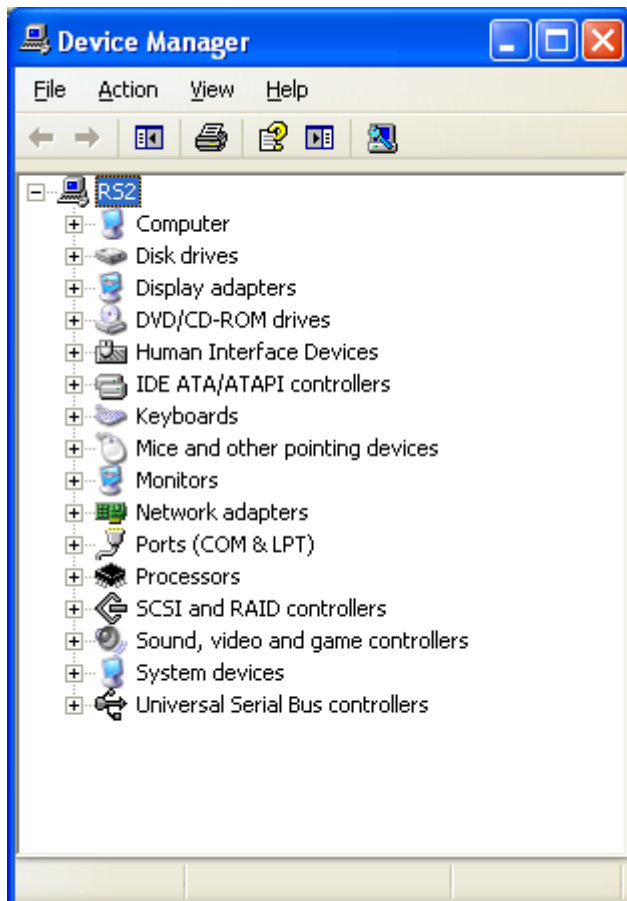


Рис. 4.3 – Окно «Диспетчер устройств»

В этом окне необходимо выбрать параметр «Ports (COM & LPT)». После выполнения описанных выше действий можно определить номер виртуального последовательного порта, реализованного через порт USB (рис. 4.4).

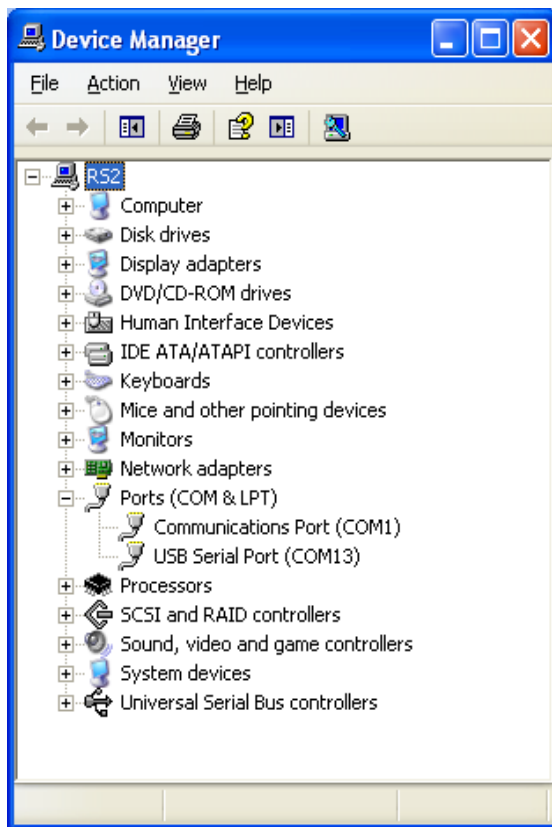


Рис. 4.4 – Определение номера «USB Serial PORT»

В приведенном примере номер порта – 13. Это номер порта необходимо использовать при разработке программного обеспечения, использующего виртуальный последовательный порт USB. При необходимости номер виртуального последовательного интерфейса может быть модифицирован через использование функции «Свойства» данного порта.

5. ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

- Разработка устройства измерения цифрового сигнала на базе микроконтроллера Atmega16.
- Разработка устройства формирования цифрового сигнала на базе микроконтроллера Atmega16.
- Разработка устройства измерения аналогового сигнала на базе микроконтроллера Atmega16.
- Разработка устройства формирования аналогового сигнала на базе микроконтроллера Atmega16.
- Реализация устройства ввода-вывода цифровой информации.
- Разработка устройства передачи данных по последовательному каналу RS232.

В качестве цифрового сигнала используется сигнал цифрового генератора (амплитуда не более 3В). В качестве аналогового сигнала используется синусоидальный сигнал (размах сигнала не более 5В).

Параметры сигналов (амплитуда и длительность полупериодов) и алгоритмы обработки задаются для каждого обучающегося индивидуально.

6. ПРИМЕРЫ ПРОГРАММ

6.1 Программа приема и передачи цифрового сигнала

Данная программа формирует цифровой сигнал на выходе OUT1 в соответствии с входным сигналом, поступающим на вход IN1.

```
.include "m16def.inc"

.cseg
.org 0
.def TMP=R16

//описание входов и выходов порта D
// RXD = 0
// TXD = 1
.equ in1 = 2 //вход 1
.equ in2 = 3 //вход 2
.equ out1 = 4//выход 1
.equ out2 = 5//выход 2
// WR = 6
// SEL = 7

// установка направления работы порта D
// RxD, In1,In2 - вход,
// TxD, Out1,Out2,SEL,WR – выход
// используется вход in1 и выход out1
    ldi    tmp,(1<<out1)
    out    DDRD,tmp
    ldi    tmp,0b11101111
    out    PORTD,tmp

m1:
// опрос входного сигнала in1
```

```
    sbis    PIND,in1
    jmp m2
// установка выходного сигнала out1 в 1
    sbi     PORTD,out1
    jmp m1
m2:
// установка выходного сигнала out1 в 0
    cbi     PORTD,out1
jmp m1
```

6.2 Программа формирования цифрового сигнала с программным опросом окончания работы таймера

Данная программа формирует цифровой сигнал на выходе OUT1 с параметрами 20 мксек - длительность сигнала и 20+40 мксек – период сигнала. Используется нормальный режим работы (0b0000) таймера 1 с программным опросом момента сравнения содержимого таймера с регистрами сравнения: В - длительность сигнала, А – период сигнала. Перегрузка работы таймера 1 осуществляется в основном цикле программно.

```
.include "m16def.inc"

.def tmp      = r16
.def r_zero   = r18

//входы и выходы порта D
// RXD = 0
// TXD = 1
.equ in1 = 2 //вход 1
.equ in2 = 3 //вход 2
.equ out1 = 4//выход 1
.equ out2 = 5//выход 2
// WR = 6
// SEL = 7

//частота процессора 11.0592 МГц
.equ CPU_FIO = 11

//коэффициент делителя
.equ PRS = 0b010
//коэффициент деления частоты
.equ PRC = 8

//параметры формируемого сигнала
.equ T_H = 20
```

```

.equ T_L = 40
.equ Time_H = (T_H * CPU_FIO)/PRC
.equ Time_L = (T_L * CPU_FIO)/PRC

.cseg
//вектора прерываний
.org 0
rjmp RESET

//адреса прерываний таймера по сравнению( не используются)
.org OC1Aaddr
.org OC1Baddr

reset:
// установка стека в конец памяти RAM
    ldi    tmp,high(RAMEND)
    out    SPH,tmp
    ldi    tmp,low (RAMEND)
    out    SPL,tmp

// установка направления работы порта D
// RxD,   In1,In2 - вход,
//      TxD,   Out1,Out2,SEL,WR - выход
//используется только выход 1
    ldi    tmp,(1<<out1)
    out    DDRD,tmp
    ldi    tmp,$FF
    out    PORTD,tmp

//начальное выключение сигнала
    cbi    PORTD,out1

//загрузка регистра сравнения COMPA
//период сигнала
    ldi    tmp,high(Time_H+Time_L)
    out    OCR1AH,tmp

```

```
ldi    tmp,low(Time_H+Time_L)
out    OCR1AL,tmp
```

//загрузка регистра сравнения COMPB

//длительность импульса

```
ldi    tmp,high(Time_H)
out    OCR1BH,tmp
ldi    tmp,low(Time_H)
out    OCR1BL,tmp
```

//установка начального значения таймера

```
clr    r_zero
out    TCNT1H, r_zero
out    TCNT1L, r_zero
```

//задание режима работы таймера

//и коэффициента делителя записи CS0-CS2

//запуск таймера

```
ldi    tmp,PRS
out    TCCR1B,tmp
ldi    tmp,0
out    TCCR1A,tmp
```

//Основной цикл

MainLoop:

//включение сигнала

```
sbi    PORTD,out1
```

//опрос прерывания по COMPA

//длительность импульса

Rise:

```
in     tmp,TIFR
sbrs   tmp,OCF1B
rjmp   Rise
```

//сброс флага прерывания

```
ldi    tmp,(1<<OCF1B)
```

```
out    TIFR,tmp
```

```
//выключение сигнала
```

```
cbi    PORTD,out1
```

```
//опрос прерывания по COMPB
```

```
//период сигнала
```

```
Fall:
```

```
in     tmp,TIFR
```

```
sbrs   tmp, OCF1A
```

```
rjmp   Fall
```

```
//сброс флага прерывания
```

```
ldi     tmp,(1<<OCF1A)
```

```
out     TIFR, tmp
```

```
//запись начального значения таймера
```

```
out     TCNT1H, r_zero
```

```
out     TCNT1L, r_zero
```

```
rjmp    MainLoop
```

6.3 Программа формирования цифрового сигнала с работой таймера по прерыванию

Данная программа формирует цифровой сигнал на выходе OUT1 с параметрами 40 мксек - длительность сигнала и 40+60 мксек – период сигнала. Используется режим работы таймера 1 с прерываниями по сравнению СТС (0b0100). Это режим использует прерывание по результату сравнения содержимого таймера с регистрами сравнения: В- длительность сигнала, А – период сигнала. Перезагрузка работы таймера осуществляется автоматически при сравнении содержимого таймера 1 с регистром сравнения А.

```
.include "m16def.inc"

.def tmp      = r16
.def r_zero   = r18

// входы и выходы порта D
// RXD = 0
// TXD = 1
.equ in1  = 2 //вход 1
.equ in2  = 3 //вход 2
.equ out1 = 4//выход 1
.equ out2 = 5//выход 2
// WR = 6
// SEL = 7

//частота процессора
.equ CPU_FIO = 11

//код режима предделителя
.equ PRS = 0b010
//коэффициент деления частоты предделителя
.equ PRC = 8

//параметры формируемого сигнала
```

```

.equ T_H = 40
.equ T_L = 60
.equ Time_H = (T_H * CPU_FIO)/PRC
.equ Time_L = (T_L * CPU_FIO)/PRC

.cseg
//вектора прерываний
.org 0
rjmp RESET

.org OC1Aaddr
rjmp TMR1_COMPA
.org OC1Baddr
rjmp TMR1_COMPB

reset:
// установка стека в конец памяти RAM
    ldi    tmp,high(RAMEND)
    out    SPH,tmp
    ldi    tmp,low (RAMEND)
    out    SPL,tmp

// установка направления работы порта D
// RxD,  In1,In2 - вход,
//  TxD,   Out1,Out2,SEL,WR - выход
//используется только выход 1
    ldi    tmp,(1<<out1)
    out    DDRD,tmp
    ldi    tmp,$FF
    out    PORTD,tmp

//начальное выключение сигнала
    cbi    PORTD,out1

//загрузка регистра сравнения COMPA
//период

```

```
ldi    tmp,high(Time_H+Time_L)
out     OCR1AH,tmp
ldi     tmp,low(Time_H+Time_L)
out     OCR1AL,tmp
```

//загрузка регистра сравнения COMPB

//длительность импульса

```
ldi     tmp,high(Time_H)
out     OCR1BH,tmp
ldi     tmp,low(Time_H)
out     OCR1BL,tmp
```

//разрешение режима SLEEP

```
ldi     tmp, (1<<SE)
out     MCUCR,tmp
```

//разрешение прерывания по сравнению

```
ldi     tmp, (1<<OCIE1B) | (1<<OCIE1A)
out     TIMSK, tmp
```

//разрешение прерываний

```
sei
```

//установка начального значения таймера

```
clr     r_zero
out     TCNT1H, r_zero
out     TCNT1L, r_zero
```

//задание режима сброса таймера при совпадении COMPA

//и коэффициента предделителя WGM12, CS0-CS2

//запуск таймера

```
ldi     tmp,(1<<WGM12) | PRS
out     TCCR1B,tmp
ldi     tmp,0
out     TCCR1A,tmp
```

//Основной цикл

MainLoop:

//включение сигнала

sbi PORTD,out1

sleep

//выключение сигнала

cbi PORTD,out1

sleep

rjmp MainLoop

//обработка прерывания таймера по сравнению COMPA

TMR1_COMPA:

reti

//обработка прерывания таймера по сравнению COMPB

TMR1_COMPB:

reti

6.4 Программа приема и передачи данных по каналу RS232 с использованием ожидания готовности

Данная программа принимает данные от персонального компьютера и возвращает данные по каналу RS232. На компьютере прием и передача данных осуществляется с помощью терминальной программы ZOC. Обмен осуществляется с опросом готовности.

```
.include "m16def.inc"
.cseg
.org 0
.def TMP=R16
    rjmp Reset

Reset:
// установка стека в конец памяти RAM
    ldi tmp,high(RAMEND)
    out SPH,tmp
    ldi tmp,low (RAMEND)
    out SPL,tmp

// программирование последовательного порта
// частота процессора
.equ CPU_HZ = 11059200
//скорость приёма и передачи
.equ BAUD=9600

//значение регистра скорости USART
.equ UBRR_VAL = (CPU_HZ / 16 / BAUD) - 1
.equ UBRR_H = high(UBRR_VAL)
.equ UBRR_L = low(UBRR_VAL)

//установка регистров скорости
//сначала старший
    ldi tmp,UBRR_H
    out UBRRH,tmp
```

```

//потом младший
    ldi tmp,UBRR_L
    out UBRRL,tmp

//8 бит данных, 2 стоп-бита
    ldi tmp, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
    out UCSRC,tmp

//разрешение работы приёмника и передатчика
    ldi tmp, (1<<RXEN)|(1<<TXEN)|(1<<RXCIE)
    out UCSRB,tmp

//Основной цикл
MainLoop:

//здесь можно выполнять фоновую задачу

//проверяем, не принят ли символ
USART_Receive:
    sbis UCSRA,RXC
    rjmp MainLoop

//получаем символ
    in tmp, UDR

//изменяем его +1
    inc tmp

//ожидание завершения прошлой передачи
USART_Transmit:
    sbis UCSRA,UDRE
    rjmp USART_Transmit

//передача символа
    out UDR,tmp
    rjmp MainLoop

```

6.5 Программа приема и передачи данных по каналу RS232 с использованием прерываний

Данная программа принимает данные от персонального компьютера и возвращает данные по каналу RS232. На компьютере прием и передача данных осуществляется с помощью терминальной программы ZOC. Обмен осуществляется с опросом готовности.

```
.include "m16def.inc"

//рабочий регистр
.def tmp    = r16

.cseg
.org 0
.def TMP=R16
    rjmp Reset

//адрес прерывания последовательного интерфейса
.org URXCaddr
    rjmp URXC_Int

Reset:
// установка стека в конец памяти RAM
    ldi tmp,high(RAMEND)
    out SPH,tmp
    ldi tmp,low (RAMEND)
    out SPL,tmp

//программирование последовательного порта
//частота процессора
.equ CPU_HZ = 11059200
//скорость приёма и передачи
.equ BAUD=9600

//значение регистра скорости USART
.equ UBRR_VAL = (CPU_HZ / 16 / BAUD) - 1
```

```

.equ UBRR_H = high(UBRR_VAL)
.equ UBRR_L = low(UBRR_VAL)

//установка регистров скорости
//начала старший
    ldi tmp,UBRR_H
    out UBRRH,tmp
//потом младший
    ldi tmp,UBRR_L
    out UBRR_L,tmp

//8 бит данных, 2 стоп-бита
    ldi tmp, (1<<URSEL)|(1<<USBS)|(3<<UCSZ0)
    out UCSRC,tmp

//разрешение работы приёмника и передатчика, прерываний приёмника
    ldi tmp, (1<<RXEN)|(1<<TXEN)|(1<<RXCIE)
    out UCSRB,tmp

//разрешение глобальных прерываний
    sei

//Основной цикл
MainLoop:
//здесь можно выполнять фоновую задачу
    rjmp MainLoop

//Обработчик прерывания по приёму символа
URXC_Int:
//сохраняем рабочий регистр и флаги
    push tmp
    in tmp,SREG
    push tmp

//получаем символ
    in tmp,UDR

```

//изменяем его +1

inc tmp

//передаём символ. скорости приёма и передачи одинаковые,

//поэтому готовность передатчика не проверяем

out UDR, tmp

//восстанавливаем флаги и рабочий регистр

pop tmp

out SREG,tmp

pop tmp

//возврат из прерывания

reti

7. ТЕСТОВАЯ ПРОГРАММА

На компьютере представлена тестовая программа TEST, позволяющая проверить основные функции лабораторного макета.

```
.include "m16def1.inc"           ;описание регистров микроконтроллера
.def tmp    = r16
.cseg
.org 0
//вектора прерываний
rjmp RESET
;rjmp EXT_INT0
;rjmp EXT_INT1
;rjmp TIM2_COMP
;rjmp TIM2_OVF
;rjmp TIM1_CAPT

.org OC1Aaddr                 ;Output Compare1A Interrupt Vector Address
rjmp TIM1_COMPTA              ; вектор прерывания таймера 1 по сравнению
;rjmp TIM1_COMPTB
;rjmp TIM1_OVF
;rjmp TIM0_OVF
;rjmp SPI_STC
;rjmp USART_RXC
;rjmp USART_UDRE
;rjmp USART_TXC

.org ADCCaddr                 ;ADC Interrupt Vector Address
rjmp ADC_TO_DAC               ; вектор прерывания от АЦП
;rjmp EE_RDY
;rjmp ANA_COMP
;rjmp TWSI
;rjmp EXT_INT2
;rjmp TIM0_COMP
;rjmp SPM_RDY
```

RESET:

// установка стека в конец памяти RAM

ldi tmp,high(RAMEND)

out SPH,tmp

ldi tmp,low (RAMEND)

out SPL,tmp

// установка направления работы порта A - входы

ldi tmp,0

out DDRA,tmp

ldi tmp,\$FF

out PORTA,tmp

// установка направления работы порта B

//I_T1 - вход, O_T1 - выход

ldi tmp,0b00001000

out DDRB,tmp

ldi tmp,\$FF

out PORTB,tmp

// установка направления работы порта C - выходы

ldi tmp,\$FF

out DDRC,tmp

out PORTC,tmp

// установка направления работы порта D

// RxD, In1,In2 - вход,

// TxD, Out1,Out2,SEL,WR - выход

ldi tmp,0b11110010

out DDRD,tmp

ldi tmp,\$FF

out PORTD,tmp

// Настройка АЦП

ldi tmp,\$40

```

    out MCUCR,tmp
// Канал 0 - Выравнивание влево результата
    ldi tmp,$20
    out ADMUX,tmp
// АЦП - включен, одиночное преобразование
// прерывание от компаратора частота 78 кГц
    ldi tmp,$88
    out ADCSR,tmp
// Разрешение прерывания
    sei
    sbi ADCSR,ADSC
    por por por por por    Первое одиночное преобразование

```

При нажатии кнопки «КН 1» проверяется работа канала АЦП-ЦАП. На вход «АЦП» макета необходимо подать сигнал с аналогового генератора с размахом не более 5В. Выход «ЦАП 1» подключить к осциллографу. На экране осциллографа можно наблюдать обработанный АЦП и ЦАП входной сигнал.

```

// Режим 1. Работа с АЦП.
// Получение отсчёта с АЦП
// и вывод его в ЦАП в обработке прерывания АЦП.
prg1:
    sei
    sbi ADCSR,ADSC    ; Запуск АЦП
    sleep             ; Sleep, пока идет преобразование АЦП
    ; переход на вывод в ЦАП в конце преобразования
    por por por por por

// проверка нажатия кнопок
    sbis PINA,4
    rjmp prg4         ; Переход по нажатию кнопки 4
    sbis PINA,5
    rjmp prg3         ; Переход по нажатию кнопки 3
    sbis PINA,6
    rjmp prg2         ; Переход по нажатию кнопки 2

```

rjmp prg1

При нажатии кнопки «КН 2» проверяется работа канала обработки цифрового сигнала. На вход «ВХ1» или «ВХ2» макета необходимо подать сигнал с цифрового генератора. Выход «ВЫХ1» или соответственно «ВЫХ2» подключить к осциллографу. На экране осциллографа появится цифровой сигнал соответствующий по параметрам входному.

//Режим 2. Работа с цифровым сигналом.

// Прием сигнала со входа IN1 и передача на выход OUT1

// IN2 - OUT2

prg2:

cli ; Выключение прерываний

sbic PIND,2

sbi PORTD,4 ; Если In1=0, то Out1=0

sbic PIND,3

sbi PORTD,5 ; Если In2=0, то Out2=0

sbis PIND,2

cbi PORTD,4 ; Если In1=1, то Out1=1

sbis PIND,3

cbi PORTD,5 ; Если In2=1, то Out2=0

sbis PINA,4

rjmp prg4 ; Переход по нажатию кнопки 4

sbis PINA,5

rjmp prg3 ; Переход по нажатию кнопки 3

sbis PINA,7

rjmp prg1 ; Переход по нажатию кнопки 1

rjmp prg2

При нажатии кнопки «КН 3» проверяется работа таймера. Выход «ВЫХ3» подключить к осциллографу. На экране осциллографа появится цифровой сигнал сформированный с помощью таймера длительностью 50 мсек.

// Режим 3. Работа с таймером.

// Формирование меандра частотой 50 Гц

prg3:

cli

// Настройка таймера

// Разрешение прерывания от Comp1A по сравнению

ldi tmp,0b00010000

out TIMSK,tmp

// Загрузка в OCR1AH и OCR1AL коэффициента счета

// При тактовой частоте 11.0592 МГц и коэффициенте делителя 1024

// коэффициент деления **006Ch** (108d) соответствует выходной частоте 50 Гц

ldi tmp,\$00

out OCR1AH,tmp

ldi tmp,\$6C

out OCR1AL,tmp

// Запуск таймера с частотой fclk/1024

ldi tmp,0b00000000

out TCCR1A,tmp

ldi tmp,0b00000101

out TCCR1B,tmp

sei

prg31:

sbis PINA,4

rjmp prg3_4 ; Переход по нажатию кнопки 4

sbis PINA,5

rjmp prg3_2 ; Переход по нажатию кнопки 2

sbis PINA,7

rjmp prg3_1 ; Переход по нажатию кнопки 1

rjmp prg31

prg3_4:

cli

```
ldi tmp,0
out TCCR1B,tmp ; Остановка таймера
rjmp prg4
```

```
prg3_2:
cli
ldi tmp,0
out TCCR1B,tmp
rjmp prg2
```

```
prg3_1:
cli
ldi tmp,0
out TCCR1B,tmp
rjmp prg1
```

При нажатии кнопки «КН 4» формируется тестовый сигнал ЦАП. На экране осциллографа можно наблюдать пилообразный сигнал на выходе «ЦАП1» и пилообразный сигнал противоположной полярности – на выходе «ЦАП2».

// Режим 4. Формирование пилообразного напряжения

Prg4:

```
cli

inc tmp ; формирование пилообразного напряжения
cbi PORTD,6 ; Включаем сигнал записи WR в ЦАП
out PORTC,tmp
sbi PORTD,7 ; Включаем сигнал SEL для ЦАП А (1)- канал В
cbi PORTD,7 ; Включаем сигнал SEL для ЦАП В (0)- канал А
sbi PORTD,6 ; Выключаем сигнал записи WR в ЦА

sei
sbis PINA,5
rjmp prg3 ; Переход по нажатию кнопки 3
sbis PINA,6
rjmp prg2 ; Переход по нажатию кнопки 2
```

```
sbis PINA,7
```

```
rjmp prg1 ; Переход по нажатию кнопки 1
```

```
rjmp prg4
```

```
// Процедура обслуживания прерывания от АЦП
```

```
// для записи в ЦАП полученных данных
```

```
ADC_TO_DAC:
```

```
in tmp,ADCH ; В tmp - значение из АЦП
```

```
cbi PORTD,6 ; Включаем сигнал записи WR в ЦАП
```

```
out PORTC,tmp
```

```
// (опорное +U, выходное -U)
```

```
sbi PORTD,7 ; Включаем сигнал SEL для ЦАП А (1)- канал В
```

```
// (опорное -U, выходное +U)
```

```
cbi PORTD,7 ; Включаем сигнал SEL для ЦАП В (0)- канал А
```

```
sbi PORTD,6 ; Выключаем сигнал записи WR в ЦАП
```

```
reti
```

```
// Процедура обслуживания прерывания от таймера
```

```
// переключение выходного сигнала OUT2
```

```
TIM1_COMPTA:
```

```
cli
```

```
in tmp,PINB ; ввод данных с порта В
```

```
andi tmp,0b00001000 ; маскирование бита PB.3
```

```
com tmp ; инверсия бита
```

```
out PORTB,tmp ; вывод данных
```

```
reti
```

СПИСОК ЛИТЕРАТУРЫ

1. Микроконтроллеры семейства AVR. -
<http://www.gaw.ru/html.cgi/txt/doc/micros/avr/index.htm>
2. Хартов В.Я. Микроконтроллеры AVR. - Москва: МГТУ им. Баумана, 2007. - 242 с.
3. Белов А.В. Создаем устройства на микроконтроллерах – Санкт-Петербург: Наука и техника, 2007. - 295 с.
4. Голубцов М.С. Микроконтроллеры AVR. От простого к сложному. – Москва: -Солон-Пресс 2003. - 286 с.