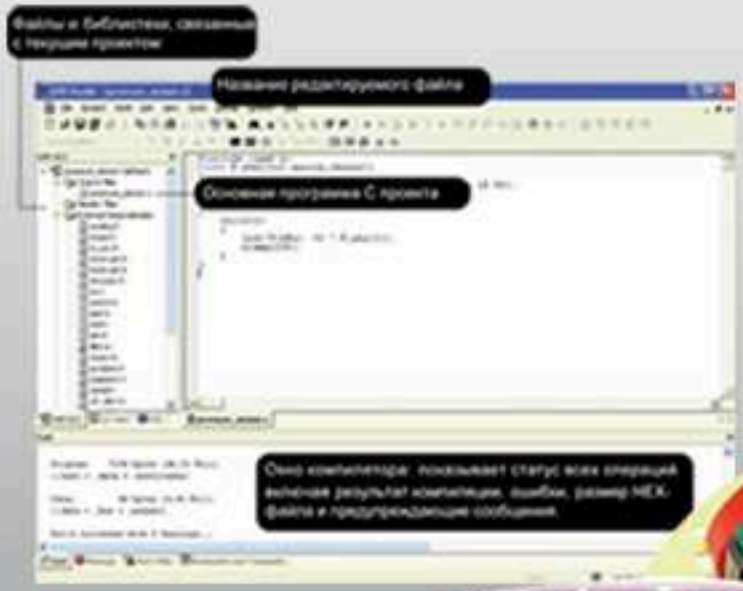
Набор для построения робота

MicroCamp2.0

на основе микроконтроллера ATmega8

Набор MicroCamp предназначен для изучения основ программирования современных микроконтроллеров путем построения робота.

Выполнение предложенных заданий позволит освоить программирование микроконтроллера и изучить принципы его взаимодействия с датчиками и исполнительными механизмами.



Набор включает в себя
плату микроконтроллера,
контактный модуль,
инфракрасный отражательный датчик
мотор постоянного тока с редуктором
и много других частей для создания и программирования робота.

Инструкция по сборке и программированию

www.inexglobal.com

inex
INNOVATIVE EXPERIMENT

Набор для построения робота

MicroCamp2.0

на основе микроконтроллера ATmega8

Набор MicroCamp предназначен для изучения основ программирования современных микроконтроллеров путем построения робота.

Выполнение всех предложенных заданий позволит освоить программирование микроконтроллера и изучить принцины его взаимодействия с датчиками и исполнительными механизмами.

Набор включает в себя плату микроконтроллера, контактный модуль, инфракрасный отражательный датчик, мотор постоянного тока с редуктором и много других частей для создания программируемого робота.

Содержание

Глава 1

Состав набора MicroCamp

Набор MicroCamp предназначен для изучения основ программирования современных микроконтроллеров путем построения робота. Выполнение всех предложенных заданий позволит освоить программирование микроконтроллера и изучить принципы его взаимодействия с датчиками и исполнительными механизмами.

Набор включает в себя плату микроконтроллера (далее по тексту “плата MicroCamp”), контактный модуль, инфракрасный отражательный датчик, мотор постоянного тока с редуктором и много других частей для создания программируемого робота.

На рис. 1-1 показано расположение элементов на плате контроллера MicroCamp.

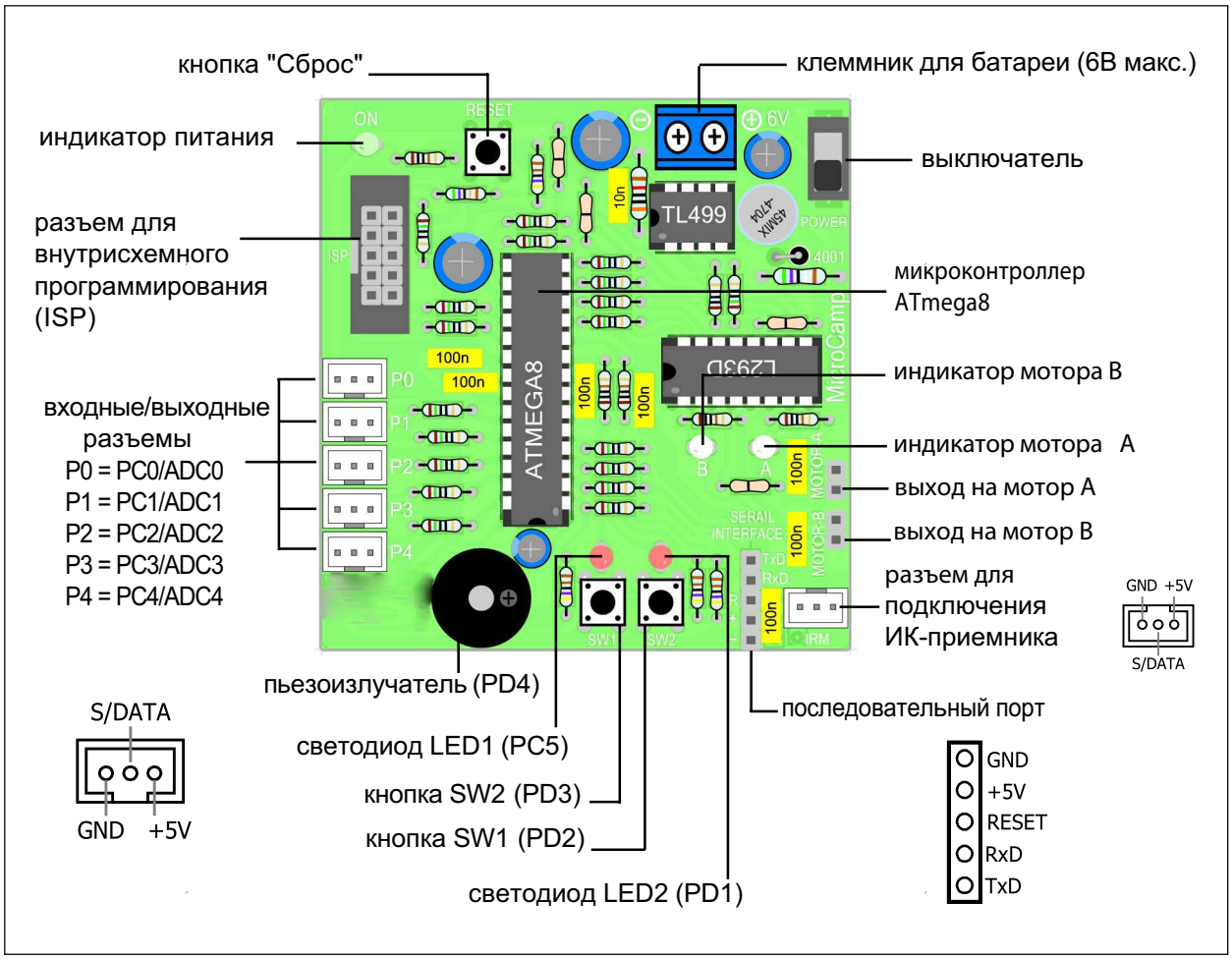


Рисунок 1-1 Расположение элементов на плате MicroCamp

1.1 Состав набора MicroCamp

1.1.1 Плата контроллера MicroCamp

- В качестве основного микроконтроллера используется 8-разрядный AVR-микроконтроллер от фирмыAtmel; ATmega8.

Он отличается следующими особенностями: 10-разрядный АЦП (ADC), флеш-память для записи программ 8KB (10,000 циклов перезаписи), флеш-память для записи данных 512 байт и ОЗУ (RAM) 512 байт.

- Частота процессора 16MHz, стабилизирована кварцевым резонатором .
- 5-канальный программируемый порт (каждый канал снабжен 3-контактным разъемом).

Каждый канал может быть запрограммирован какцифровой вход, цифровой выход или аналоговый вход. К контактам разъема подведено питающее напряжение (обычно +5В), аналоговый сигнал или цифровые данные и "земля", соответственно.

- Зарезервированный порт для подключения ИК-приемника 38kHz. Этот порт объединен со входом приемника последовательного порта (RxD), предназначенного для подключения внешних устройств с последовательным каналом передачи данных.
- Пьезоизлучатель для воспроизведения звуков.
- 2 кнопки.
- Кнопка "Сброс".
- 2 светодиода, светящихся при подаче логической единицы.
- Двухканальный драйвер моторов постоянного тока. Параметры драйвера: от 4.5 до 6В, 600мА; светодиоды индикации активности.
- Напряжение питания от +4.8 до +6В (4 батарейки размера АА). Батарейный отсек смонтирован на задней стороне платы.
- Встроенный импульсный регулятор напряжения для стабилизации +5В и защиты от провалов при включении моторов постоянного тока.

1.1.2 Внутрисхемный программатор BPX-400

Этот программатор предназначен для загрузки кода в память программ AVR-микроконтроллера. Поддерживаются большинство AVR-микроконтроллеров.

Основные особенности:

- Подключается к последовательному порту компьютера через интерфейс RS-232. Если компьютер оснащен только портом USB, рекомендуется использовать USB/Serial-конвертер. Рекомендуемый конвертер UCON-232S.
- Микроконтроллер программируется через ISP-кабель. Поддерживаются функции: Чтение, Запись, Очистка и Защита данных.

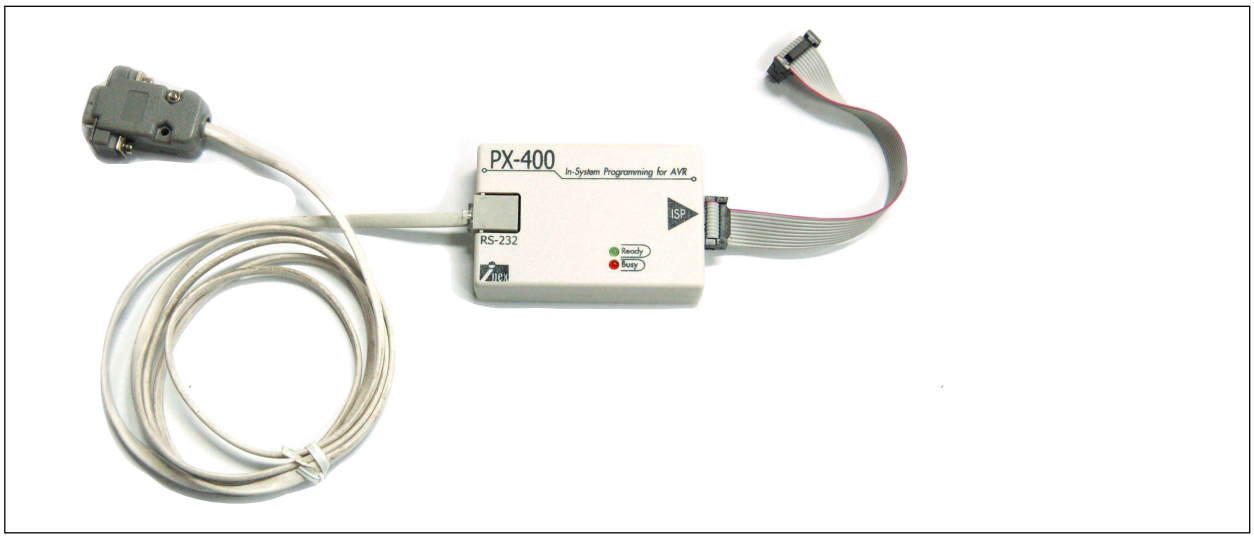


Рисунок 1-2 Внутрисхемный программатор PX-400.

- Необходимо питание +5В, поступающее с платы микроконтроллера.
- Поддерживается программой AVR Prog. Эта программа является частью AVR Studio и расположена в меню tools.

Список поддерживаемых моделей микроконтроллеров:

AT90S1200, AT90S2313, AT90S2323, AT90S2343, AT90S4433 , AT90S8515 , AT90S8535 , ATmega128 , ATmega16 , ATmega161 , ATmega162 , ATmega163 , ATmega164P , ATmega165 , ATmega168 , ATmega32 , ATmega64 , ATmega8 , ATmega8515 , ATmega8535, ATtiny12 , ATtiny13 , ATtiny15L , ATtiny2313, ATtiny26

Список моделей микроконтроллеров, поддерживаемых Avr-OSP II:

AT90CAN128, AT90CAN32, AT90CAN64 , AT90PWM2, AT90PWM3, AT90S1200, AT90S2313, AT90S2323, AT90S2343, AT90S4414, AT90S4433, AT90S4434 , AT90S8515, AT90S8515comp, AT90S8535, AT90S8535comp , ATmega103, ATmega103comp, ATmega128, ATmega1280, ATmega1281, ATmega16, ATmega161, ATmega161comp, ATmega162, ATmega163, ATmega165 , ATmega168, ATmega169 , ATmega2560, ATmega2561 , ATmega32, ATmega323, ATmega325, ATmega3250, ATmega329, ATmega3290, ATmega406, ATmega48 , ATmega64, ATmega640, ATmega644, ATmega645, ATmega6450, ATmega649 , ATmega6490 , ATmega8, ATmega8515, ATmega8535, ATmega88 , ATtiny11, ATtiny12, ATtiny13, ATtiny15, ATtiny22, ATtiny2313,ATtiny24, ATtiny25, ATtiny26, ATtiny261, ATtiny28, ATtiny44, ATtiny45, ATtiny461, ATtiny84, ATtiny85, ATtiny861

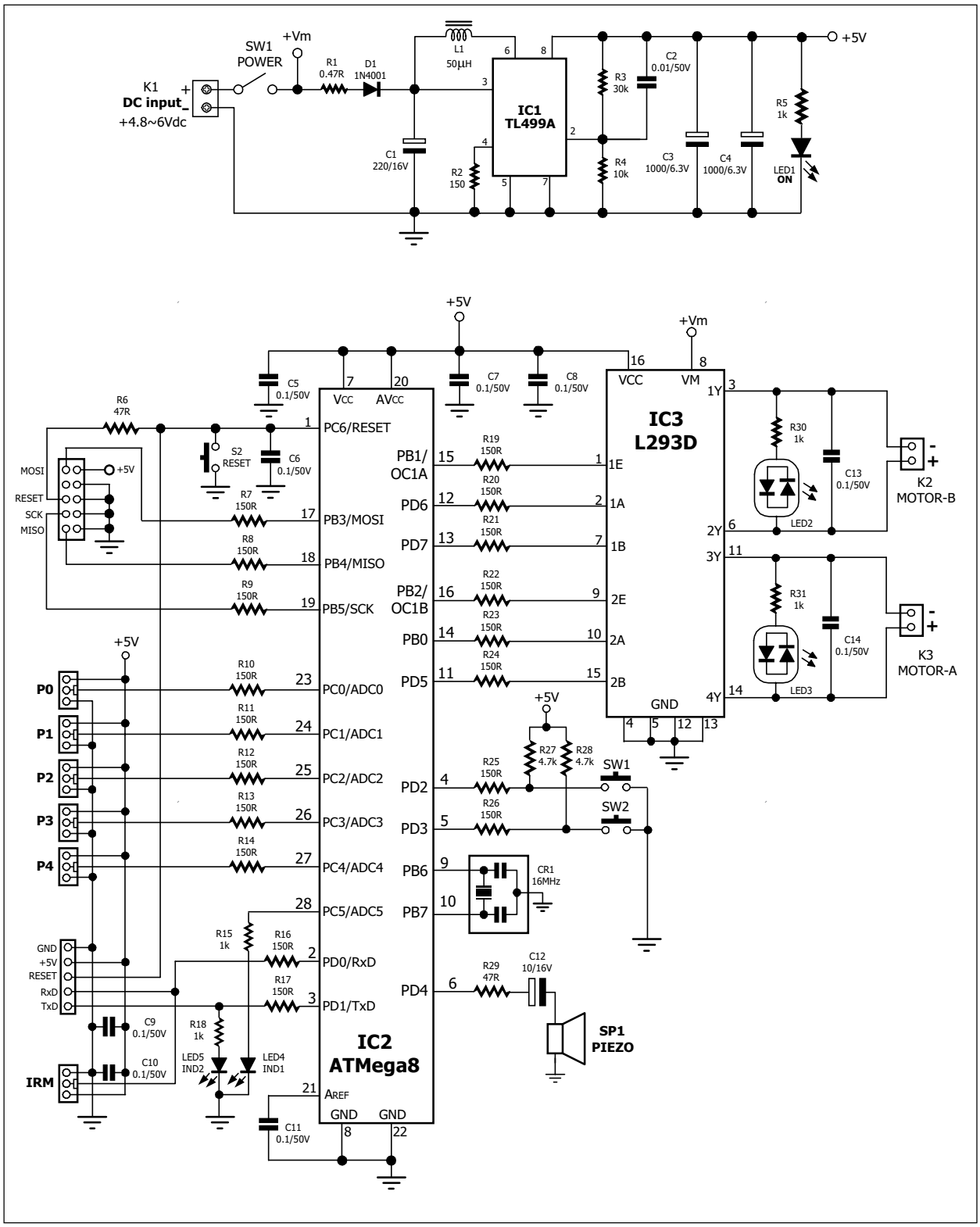


Рисунок 1-3 Схема платы MicroCamp

1.2 Описание схемы платы контроллера MicroCamp

Сердцем контроллерной платы является микроконтроллер ATmega8. Он работает с тактовой частотой 16MHz, задаваемой кварцевым резонатором, подключенным к выводам PB6 и PB7.

Порты микроконтроллера PC0 - PC4 переименованы в P0 - P4. Эти обозначения нанесены на печатную плату для упрощения монтажа. Все порты могут быть сконфигурированы как аналоговые входы или цифровые входы/выходы. Аналоговый сигнал с этих портов поступает на АЦП, встроенный в микроконтроллер ATmega8. Разрешение АЦП составляет 10-бит.

Порты PB3, PB4 и PB5 предназначены для внутрисхемного программирования. Они подключены к разъему ISP, к которому подключается программатор.

Порт PC6/RESET подключен к кнопке "Сброс" для возможности ручного перезапуска программы микроконтроллера пользователем.

Порт PD0/RxD является входом приемника последовательного порта. К этому порту подключается модуль ИК-приемника через разъем IRM и последовательный порт через 5-контактный разъем.

Порт PD1/TxD является выходом передатчика последовательного порта. Он используется для подсвечивания индикатора LED5 (обозначение на плате IND2) и подключен к выводу TxD 5-контактного разъема последовательного порта. Индикатор LED4 (IND1) подключен к порту PC5 через ограничивающий резистор.

На плате контроллера установлены две кнопки. Они подключены к портам PD2 и PD3 с подтягивающими резисторами 4.7kОм для задания уровня логической единицы при отсутствии нажатия и уровня логического нуля при нажатии кнопки.

Порт PD4 подключен через конденсатор 10мкФ к пьезоизлучателю.

На контроллерной плате смонтирован двухканальный драйвер моторов постоянного тока. Драйвер реализован на микросхеме L293D. Для управления одним мотором постоянного тока необходимо формировать три сигнала:

A и B для задания направления вращения.

E для включения соответствующего драйвера. Также можно контролировать скорость вращения мотора путем подачи ШИМ-сигнала (PWM) на этот вход. Более широкий импульс соотвествует подаче большего напряжения на мотор.

На выходе L293D включены двухцветные светодиодные индикаторы, указывающие направление вращения соответствующего двигателя. Зеленый цвет соответствует вращению вперед. Красный цвет соответствует вращению назад.

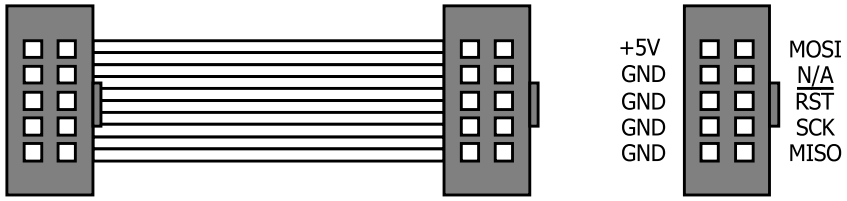
Для стабилизации питающего напряжения используется импульсный стабилизатор TL499A. Несмотря на повышенное потребление энергии при работе моторов, стабилизатор поддерживает питающее напряжение микроконтроллера на постоянном уровне +5В.

1.3 Назначение кабелей в наборе MicroCamp

Набор MicroCamp содержит некоторое количество сигнальных кабелей, предназначенных для связи между платой контроллера, программатором, модулями сенсоров и компьютером. Среди них ISP-кабель для программирования микроконтроллера, PCB3AA-8-кабели для связи с модулями датчиков и кабель для связи с компьютером.

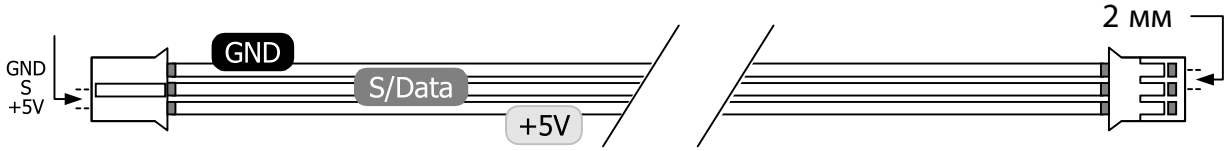
1.3.1 ISP-кабель

Кабель представляет собой 10-проводный шлейф. С обеих сторон шлейфа подключены 10-контактные разъемы IDC. Этот кабель используется для связи контроллерной платы и программатора через ISP-разъем. Кабель соответствует стандартам Atmel. Цоколевка кабеля показана на рисунке ниже.



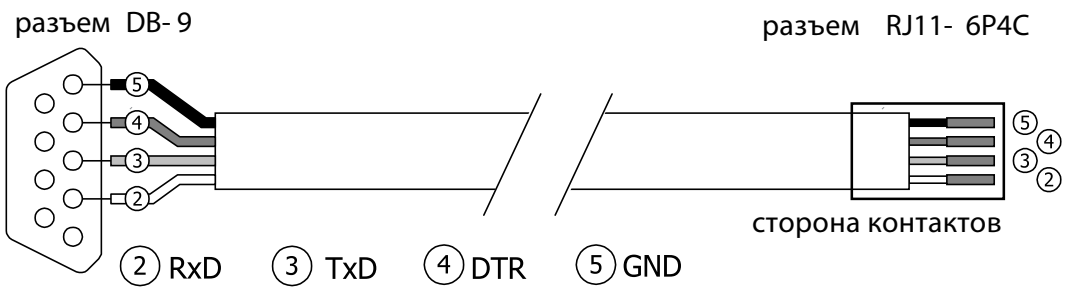
1.3.2 Кабель JST3AA-8

Это стандартный кабель INEX, трехпроводный с шагом контактов 2мм. На каждом конце установлен разъем JST. Длина кабеля 20 см. Используется для присоединения всех датчиков из набора MicroCamp к контроллерной плате. Цоколевка разъемов показана на рисунке ниже.



1.3.3 Кабель CX-4

Этот кабель используется для подключения микроконтроллерной платы к последовательному порту компьютера через интерфейс RS-232. С одной стороны кабеля подключен разъем DB-9, с другой стороны - разъем RJ-11 6P4C. Длина кабеля 1.5 метра. В наборе этот кабель используется для подключения программатора PX-400 к последовательному порту компьютера. Цоколевка кабеля показана на рисунке ниже.



1.4 Обзор микроконтроллера ATmega8

Микроконтроллер ATmega8 выполнен по технологии CMOS, 8-разрядный, микропотребляющий, основан на AVR- архитектуре RISC. Выполняя одну полноценную инструкцию за один такт, ATmega8 достигает производительности 1 MIPS на МГц, позволяя достигнуть оптимального соотношения производительности к потребляемой энергии.

В набор MicroCamp входит микроконтроллер ATmega8 в 28-выводном DIP-корпусе. Назначение выводов показано на рис. 1-4.

1.4.1 Особенности ATmega8

- Малопотребляющий 8-разрядный микроконтроллер с архитектурой AVR RISC.
- Память программ 8 Кб с возможностью перезаписать 10,000 раз, 512 байт флеш-памяти для хранения переменных (100,000 циклов перезаписи), 1 Кб ОЗУ и 32 регистра общего назначения.
- 23 порта ввода/вывода, объединенных в 3 группы.

1. Порт B (PB0 - PB7) : Два вывода (PB6 и PB7) используются для подключения кварцевого резонатора. Выводы PB2 - PB5 зарезервированы для внутрисхемного программирования. Таким образом, для общего применения остаются порты PB0 и PB1.

2. Порт C (PC0 - PC6 : 7 выводов). Порты PC0 - PC5 можно использовать в качестве аналоговых входов. PC6 обычно используется для сброса.

3. Порт D (PD0 - PD7 : 8 выводов). Этот порт можно использовать для общего применения.

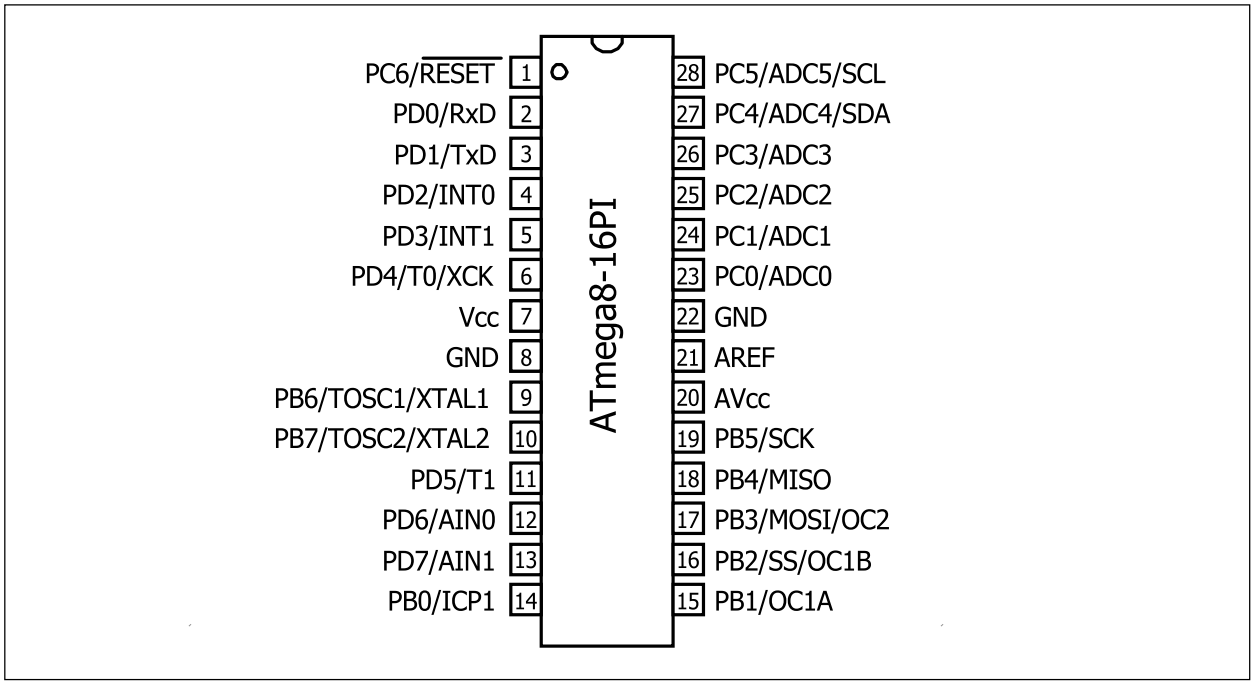


Рисунок1-4 Назначение выводов микроконтроллера ATmega8

- Два 8-разрядных Таймера/Счетчика с раздельным прескалером, режим сравнения
- 16-разрядный Таймер/Счетчик с раздельным прескалером, режим сравнения, режим захвата
- Таймер реального времени с независимым генератором
- 3 канала ШИМ
- 6 каналов 10-разрядного АЦП
- Двухпроводный последовательный интерфейс
- Программируемый последовательный USART
- Интерфейс SPI с режимами Master/Slave
- Программируемый сторожевой таймер с отдельным независимым генератором
- Встроенный аналоговый компаратор
- Сброс при включении питания, программируемая защита от провалов питания
- Встроенный калиброванный RC-генератор
- Обработка внутренних и внешних прерываний
- 5 режимов с пониженным энергопотреблением: Idle, ADC Noise Reduction, Power-save, Power-down, и Standby
- Напряжение питания 4.5 - 5.5В
- Тактовая частота 0 - 16 МГц

1.4.2 Структура микроконтроллера ATmega8

Структура микроконтроллера ATmega8 показана на рис. 1-5. Ядро процессора AVR объединяет набор RISC-инструкций с 32 регистрами. Микроконтроллер ATmega8 имеет следующие особенности 8 Кб флеш-памяти программ, 512 байт флеш-памяти переменных, 1 Кб ОЗУ, 23 порта ввода/вывода, 32 рабочих регистра общего назначения, 3 программируемых Таймера/Счетчика с режимом сравнения, внутренние и внешние прерывания, программируемый последовательный порт USART, двухпроводный последовательный интерфейс, 6-канальный АЦП с разрешением 10-бит, программируемый сторожевой таймер со встроенным генератором, последовательный порт SPI, 5 режимов пониженного энергопотребления. Режим Idle отключает процессор, при этом питание поступает на ОЗУ, Таймеры/Счетчики, порт SPI, и обслуживаются прерывания.

1.4.3 Функциональное назначение выводов ATmega8

В таблице 1-1 обобщена информация о функциональном назначении выводов ATmega8.

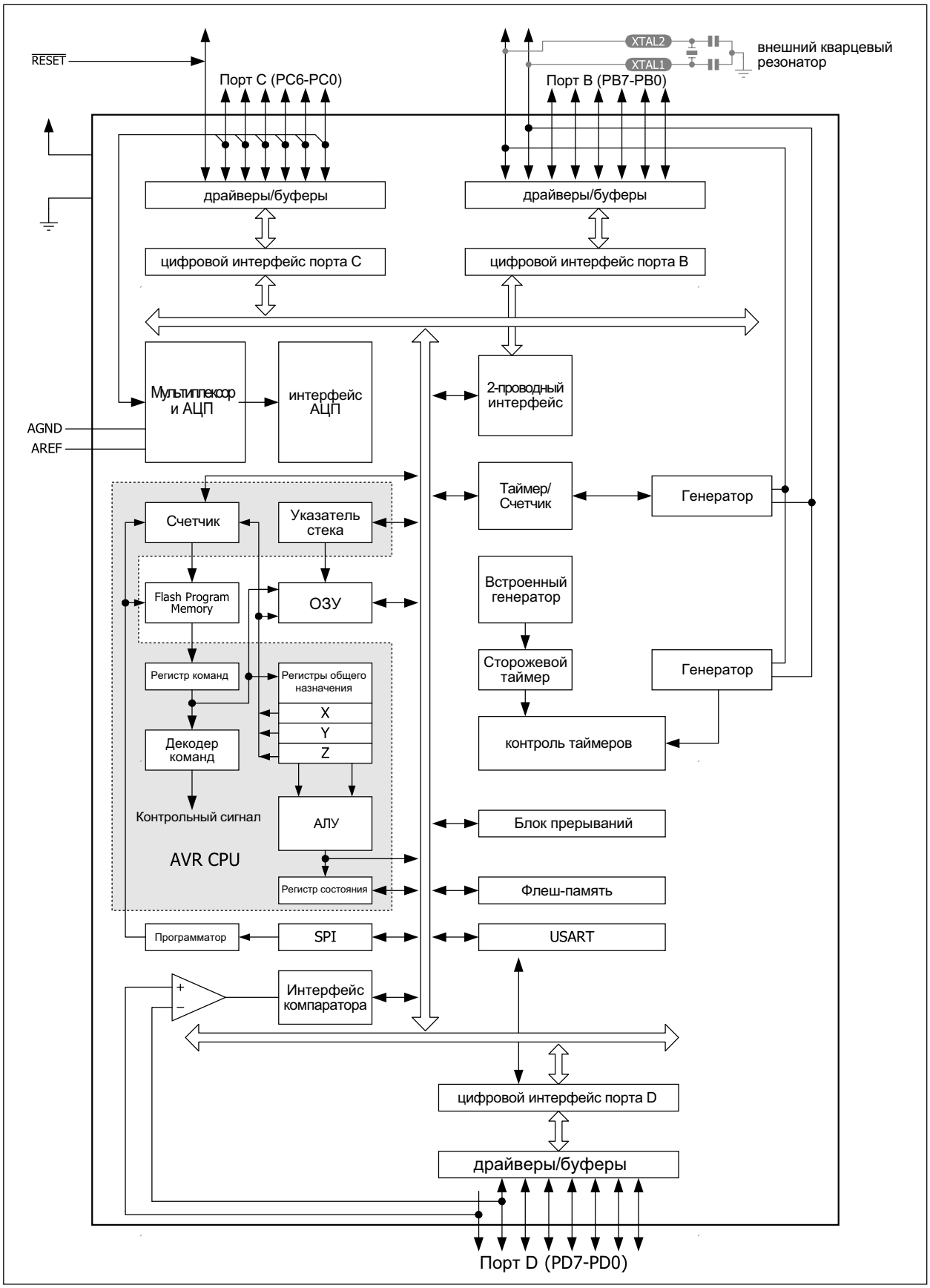


Рисунок 1-5 Структура микроконтроллера ATmega8

Название	Номер вывода	Тип	Описание
Vcc	7	Вход	- напряжение питания от +4.5 до +5.5 В
GND	8,22	Вход	- общий
AVcc	20	Вход	- напряжение питания + 5 В для модуля АЦП
AREF	21	Вход	- вход опорного напряжения для АЦП
Порт B			
Название	Номер вывода	Тип	Описание
PB0	14	Вход/Выход	- цифровой порт PB0
ICP1		Вход	- захват входа 1
PB1	15	Вход/Выход	- цифровой порт PB1
OC1A		Выход	- выход сравнения/ШИМ 1A
PB2	16	Вход/Выход	- цифровой порт PB2
OC1B		Выход	- выход сравнения/ШИМ 1B
SS		Вход	- вход Slave для SPI
PB3	17	Вход/Выход	- цифровой порт PB3
OC2		Выход	- выход сравнения/ШИМ 2
MOSI		Вход/Выход	- вход данных в режиме Slave для SPI и ISP - выход данных в режиме Master для SPI и ISP
PB4	18	Вход/Выход	- цифровой порт PB4
MISO		Вход/Выход	- вход данных в режиме Master для SPI и ISP - выход данных в режиме Slave для SPI и ISP
PB5	19	Вход/Выход	- цифровой порт PB5
SCK		Вход/Выход	- тактовый вход в режиме Slave для SPI и ISP - тактовый выход в режиме Master для SPI и ISP
PB6	9	Вход/Выход	- цифровой порт PB6 при работе от встроенного генератора
XTAL1		Вход	- тактовый вход, кварцевый или керамический резонатор
TOSC1		Вход	- не используется при работе от внешнего генератора
PB7	10	Вход/Выход	- цифровой порт PB7 при работе от встроенного генератора
XTAL2		Вход	- для подключения кварцевого или керамического резонатора
TOSC2		Выход	- тактовый выход при работе от встроенного генератора

Таблица 1-1 Назначение выводов микроконтроллера ATmega8 (начало)

Порт C			
Название	Номер вывода	Тип	Описание
PC0	23	Вход/Выход	- цифровой порт PC0
ADC0		Вход	- аналоговый вход канал 0
PC1	24	Вход/Выход	- цифровой порт PC1
ADC1		Вход	- аналоговый вход канал 1
PC2	25	Вход/Выход	- цифровой порт PC2
ADC2		Вход	- аналоговый вход канал 2
PC3	26	Вход/Выход	- цифровой порт PC3
ADC3		Вход	- аналоговый вход канал 3
PC4	27	Вход/Выход	- цифровой порт PC4
ADC4		Вход	- аналоговый вход канал 4
SDA		Вход/Выход	- канал данных для 2-проводного последовательного интрефейса
PC5	28	Вход/Выход	- цифровой порт PC5
ADC5		Вход	- аналоговый вход канал 5
SCL		Выход	- тактовый выход для 2-проводного последовательного интерфейса
PC6	1	Вход/Выход	- цифровой порт PC6
$\overline{\text{RESET}}$		Вход	- внешний сброс
Порт D			
Название	Номер вывода	Тип	Описание
PD0	2	Вход/Выход	- цифровой порт PD0
RxD		Вход	- вход приемника USART
PD1	3	Вход/Выход	- цифровой порт PD1
TxD		Выход	- выход передатчика USART
PD2	4	Вход/Выход	- цифровой порт PD2
INT0		Вход	- внешнее прерывание канал 0
PD3	5	Вход/Выход	- цифровой порт PD3
INT1		Вход	- внешнее прерывание канал 1
PD4	6	Вход/Выход	- цифровой порт PD4
XCK		Вход/Выход	- внешний такт для USART
T0		Вход	- внешний вход Timer 0
PD5	11	Вход/Выход	- цифровой порт PD5
T1		Вход	- внешний вход Timer 1
PD6	12	Вход/Выход	- цифровой порт PD6
AIN0		Вход	- вход аналогового компаратора канал 0
PD7	13	Вход/Выход	- цифровой порт PD7
AIN1		Вход	- вход аналогового компаратора канал 1

Таблица 1-1 Назначение выводов ATmega8 (окончание)

Глава 2
Программное обеспечение для набора MicroCamp

Разработка программного обеспечения для набора MicroCamp производится на языке C. В комплект поставки входят следующие программы:

1. AVR Studio : Это программное обеспечение разработано фирмой Atmel Corporation. AVR Studio это средство разработки для микроконтроллеров AVR. AVR Studio позволяет разработчику отлаживать программное обеспечение во встроенном эмуляторе процессора. AVR Studio позволяет исполнять программы на ассемблере, разработанные при помощи Atmel Corporation's AVR Assembler и программы на языке C, компилированные в WinAVR C Compiler. AVR Studio работает под управлением Microsoft Windows95 и Microsoft Windows NT. Рекомендуется версия Windows XP SP2. Эту программу можно бесплатно загрузить с сайта www.atmel.com.

2. WinAVR : WinAVR это набор утилит для компилятора C, в набор входят avrgcc (компилятор управляемый ключами), avr-libc (библиотека для avrgcc), avr-as (ассемблер), avrdude (интерфейс программатора), avarice (интерфейс JTAG ICE), avr-gdb (отладчик), programmers notepad (редактор) и другие утилиты. Все утилиты работают в среде Microsoft Windows. Обновленные версии утилиты доступны к загрузке по адресу: http://sourceforge.net/projects/winavr/.

Для программирования набора MicroCamp на языке C необходима версия WinAVR V20050214. Вначале устанавливается AVR Studio, затем WinAVR. Интерфейс AVR Studio автоматически интегрируется в WINAVR. Таким образом, разработку программ на языке C и программирование микроконтроллера производится в среде AVR Studio, которая значительно удобнее, чем WinAVR. Компиляция производится в HEX-файл, который затем загружается в память программ микроконтроллера.

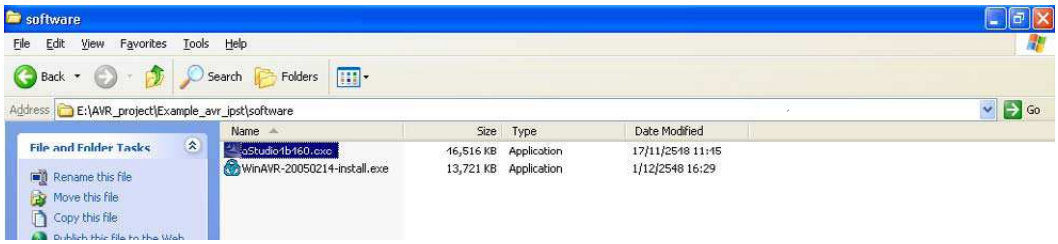
3. Библиотека: это набор дополнительных файлов, которые позволяют значительно упростить процесс разработки приложения на языке C. Например, библиотека Port control library позволяет контролировать аналоговые и цифровые входы/выходы, моторы, и т.д.

4. Программатор: это программное обеспечение позволяет загрузить .HEX-файл в микроконтроллер AVR. В этот набор включена AVRProg. Этот программатор произведен фирмой Atmel и является частью AVR Studio. Программа AVR Prog работает с программатором PX-400. Программатор PX-400 входит в состав набора MicroCamp.

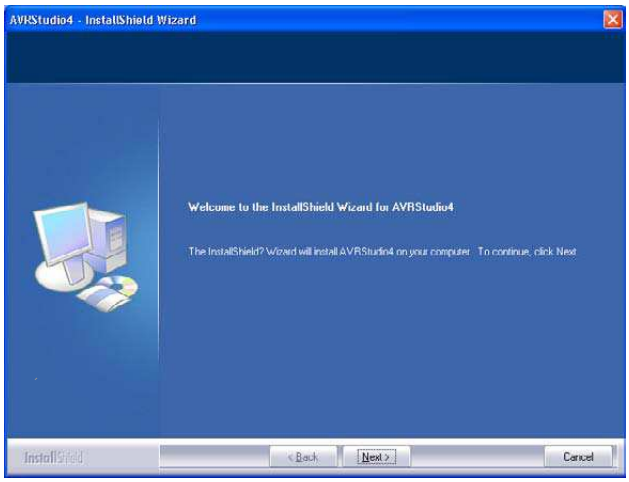
2.1 Установка AVR Studio

Установка AVR Studio под Windows XP:

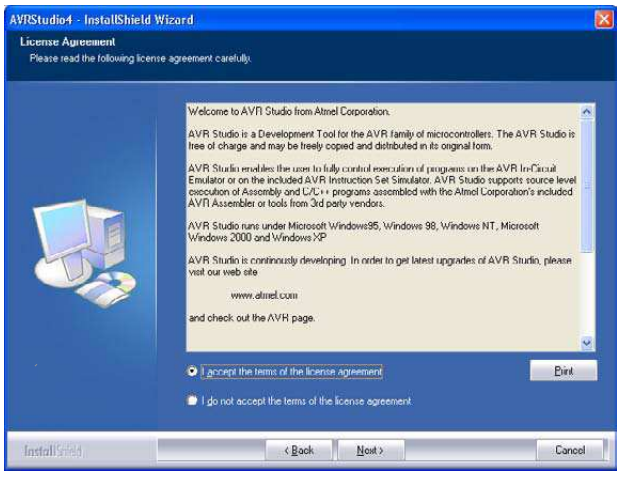
2.1.1 Запустите файл aStudio4b460.exe из директории AVR Studio на диске из набора MicroCamp.



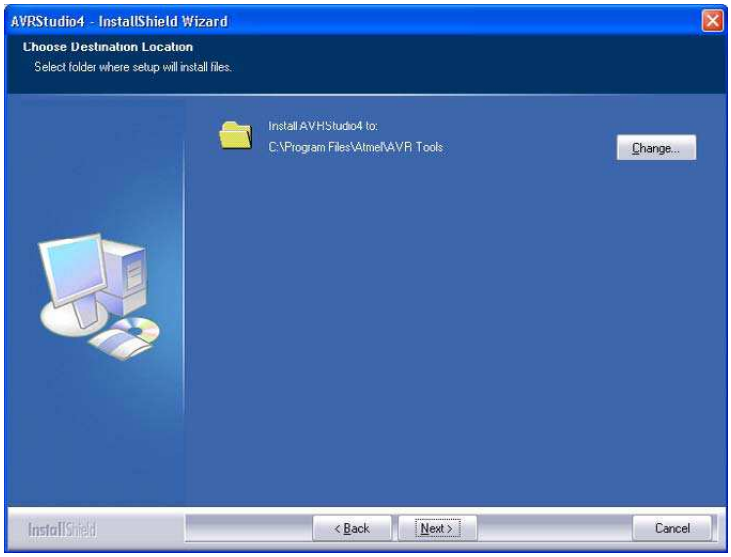
2.1.2 Запустится мастер установки. Нажмите кнопку Next для продолжения.



2.1.3 В окне лицензии выбирайте вариант: I accept the terms of the license agreement и нажимайте кнопку Next.

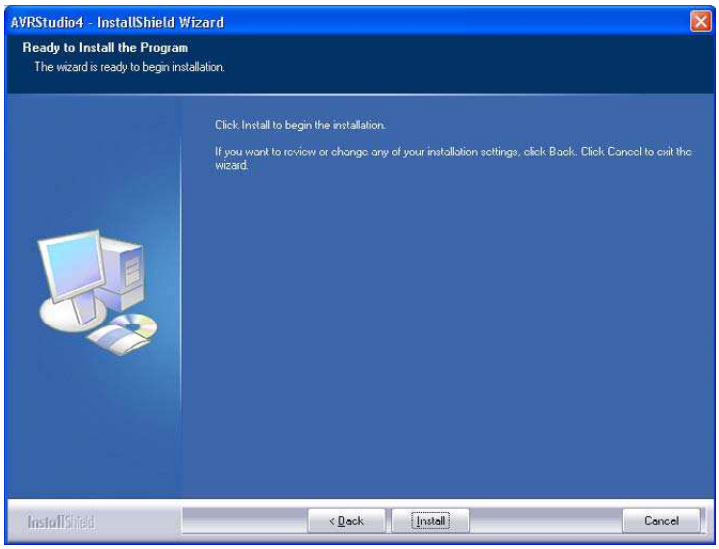


2.1.4 Появится окно выбора папки для установки. Для изменения места установки нажимайте кнопку Change и задавайте новый путь. После этого нажимайте кнопку Next.



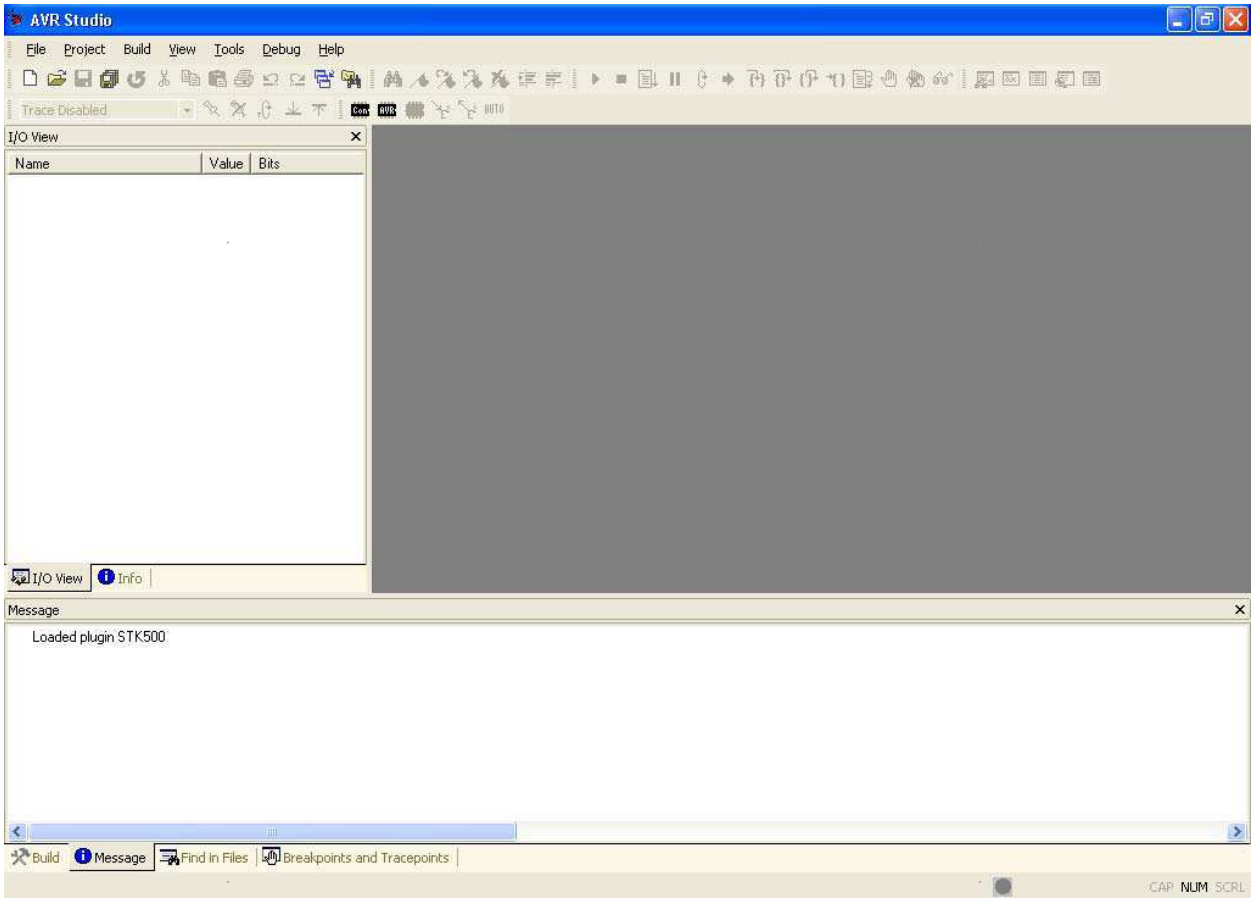
2.1.5 Появится окно Driver USB Upgrade. Нажимайте кнопку Next для пропуска этого шага.

2.1.6 Для начала установки нажмите кнопку Install.



2.1.7 После завершения установки AVR Studio, нажмите кнопку Finish.

2.1.8 Запуск программы AVR Studio. Нажимайте Пуск → Программы → Atmel AVR Tools → AVR Studio 4. Появится основное окно программы AVR Studio.

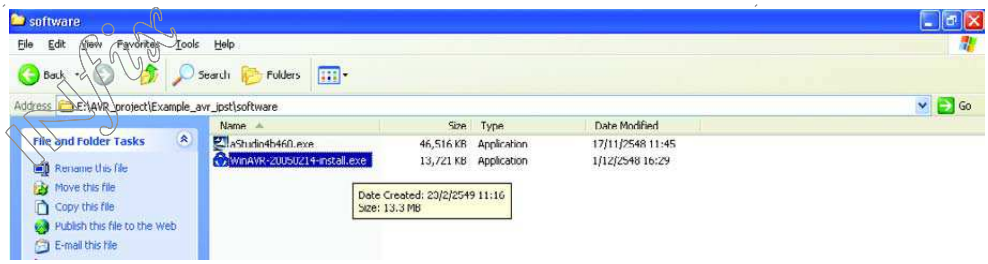


2.2 Установка WinAVR

Пожалуйста, производите установку WinAVR после установки AVR Studio. Не нарушайте последовательность действий!

Установка WinAVR под Windows XP:

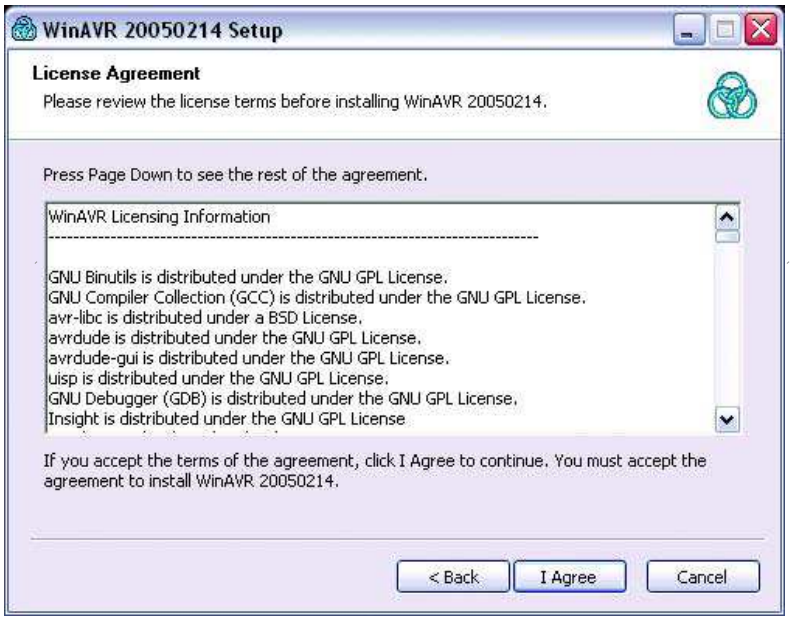
2.2.1 Запустите файл WinAVR-20050214-install.exe на диске из набора MicroCamp.



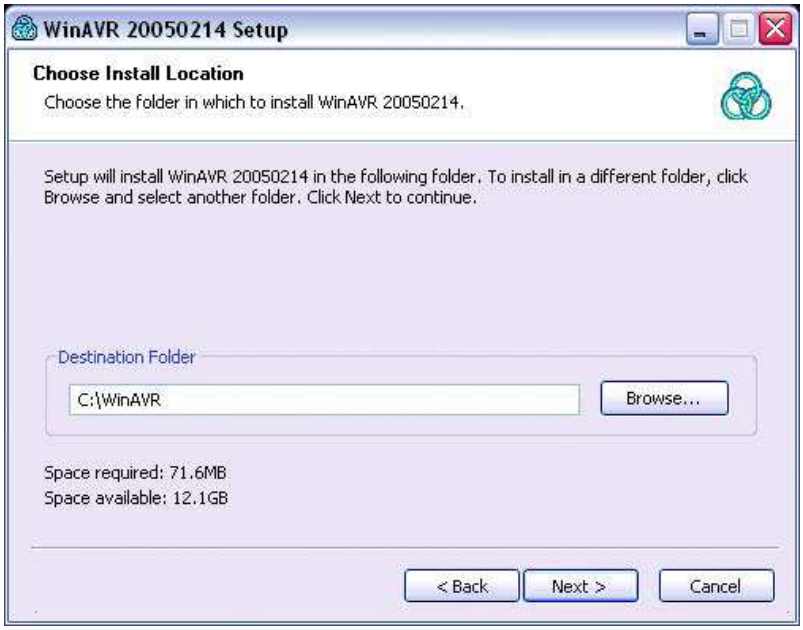
2.2.2 Первым появляется окно выбора языка установки. Выбирайте желаемый язык, затем нажмите кнопку OK.

2.2.3 Появится окно, показанное на рисунке ниже. Для продолжения нажмите кнопку Next.

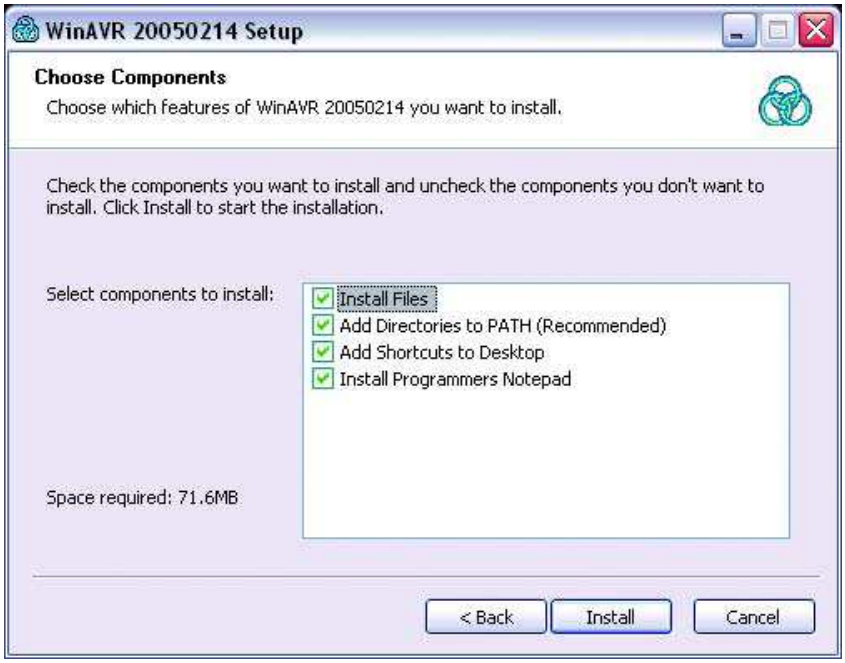
2.2.4 В окне лицензии нажимайте кнопку I agree.



2.2.5 Появится окно выбора места установки. Можно выбрать желаемое место установки путем нажатия кнопки Browse. По умолчанию программа устанавливается по адресу C:\WinAVR. Для продолжения нажимайте кнопку Next.



2.2.6 В следующем окне задаются компоненты для установки. Для начала установки нажмите кнопку Install.



2.2.7 На экране будет показываться процесс установки. Для завершения установки необходимо нажать кнопку Finish.

2.3 Копирование библиотеки

Необходимо скопировать файл библиотеки (.H файл) из директории MicroCamp_include на компакт-диске. Лучше всего поместить этот файл в директорию, в которую предполагается запись разработанных программ.

При разработке программного обеспечения для MicroCamp на AVR Studio и WinAVR, необходимо задавать путь к директории MicroCamp_include. В случае неправильно заданного пути компиляция разработанных проектов может быть невозможной.

Глава 3

Разработка программного обеспечения для набора MicroCamp на языке C при помощи AVR Studio и компилятора WinAVR

3.1 Принцип действия компилятора C

На самом деле, написание программы для микроконтроллера на языке C не означает, что именно эти коды будут загружены в память программ микроконтроллера. В память программ загружаются машинные коды, которые получаются путем компиляции кодов C специальным программным обеспечением (компилятором).

Существуют следующие этапы разработки программы:

(1) Написание программы на C при помощи текстового редактора.

(2) Компиляция кодов C в коды ассемблера микроконтроллера.

(3) Конвертирование кодов ассемблера в машинные коды (файл формата HEX).

(4) Загрузка машинных кодов в память программ микроконтроллера.

(5) Запуск микроконтроллера. Возврат к шагу 1 в случае ошибок.

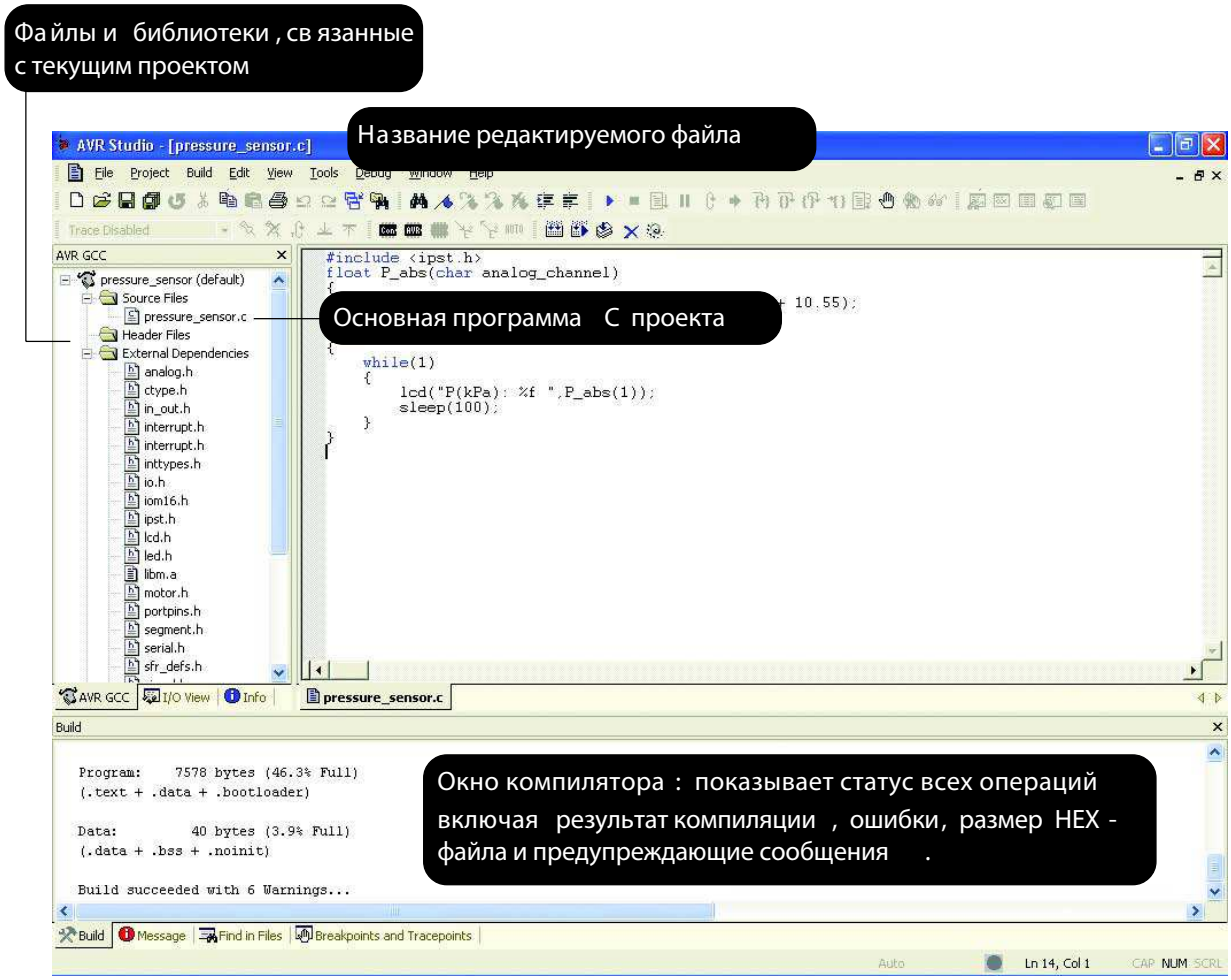
Шаги (2) и (3) не отображаются компилятором C и выполняются в фоновом режиме.
После установки AVR Studio и WINAVR необходимо скопировать библиотеку для поддержки функций MicroCamp. Файлы библиотеки MicroCamp находятся в директории MicroCamp_include на компакт-диске из набора. При разработке программ на языке C в среде AVR Studio, разработчик должен компилировать в формат файла проекта. После компиляции в HEX-файл с тем же именем, что и проект, файл загружается в память микроконтроллера ATMEGA8.

Пример:

Назовите проект test_segment. После компиляции результирующий файл будет test_segment.hex

3.2 Структура интерфейса AVR Studio V4.0

На рисунке ниже показаны основные компоненты главного окна программы AVR Studio.



3.2.1 Меню File

Включает следующие пункты:

New File	Создание пустого текстового файла
Open File	Открытие файла в текстовом редакторе или объектного файла для отладки
Close	Закрытие активного текстового файла
Save	Запись текущего файла
Save As...	Запись текущего текстового файла с новым именем
Save All	Запись всех файлов и установок проекта
Print	Печать активного текстового файла
Print Preview	Просмотр активного текстового файла
Print Setup	Установка принтера
Exit	Выход из AVR Studio с записью проекта.

3.3.2 Меню Project

Включает следующие пункты:

Project Wizard	Открыть Мастер новых проектов
	Сначала вы должны закрыть текущий проект
New Project	Создание нового проекта
	Сначала вы должны закрыть текущий проект
Open Project	Открытие проекта, или файла APS или объектного файла
Save Project	Запись текущего проекта со всеми установками
Close Project	Закрыть текущий проект
Recent Projects	Список последних проектов с возможностью загрузки
Configuration Options	Эта опция доступна только в том случае, если проект написан на ассемблере.

3.2.3 Меню Build

Включает следующие пункты:

Build	Компиляция текущего проекта
Rebuild All	Компиляция всех модулей проекта
Build and run	Компиляция и запуск в случае отсутствия ошибок
Compile	Компиляция текущего файла
Clean	Очистка текущего проекта
Export Makefile	Запись текущих установок в файл.

3.2.4 Меню Edit

Включает следующие пункты:

Undo	Отмена последнего действия (в редакторе)
Redo	Повтор последнего действия
Cut	Вырезание выделенного фрагмента
Copy	Копирование выделенного фрагмента
Paste	Вставка текста из буфера обмена в редакторе
Toggle Bookmark	Включение/выключение закладок в текущей линии
Remove Bookmarks	Удаление всех закладок
Find	Поиск в текущем файле
Find in Files	Поиск во всех файлах проекта
Next Error	Переход к следующей ошибке
Show whitespace	Показ пустых мест в проекте
Font and color	Настройка шрифтов редактора.

3.2.5 Меню View

Включает следующие пункты:

Toolbars	Подменю включения/выключения панелей инструментов
Status Bar	Включает/выключает отображение строки состояния (строка внизу экрана)
Disassembler	Включает/выключает окно дизассемблера
Watch	Включает/выключает окно просмотра
Memory	Включает/выключает просмотр памяти
Memory 2	Включает/выключает просмотр памяти 2
Memory 3	Включает/выключает просмотр памяти 3
Register	Включает/выключает просмотр регистров

3.2.6 Меню Tools

Это меню предназначено для настройки интерфейса с программаторами. AVR Studio поддерживает большое количество программаторов. Для набора MicroCamp необходимо выбирать AVRprog. Это программное обеспечение для программатора PX-400.

Необходимо подключать PX-400 к COM-порту перед запуском AVRprog.

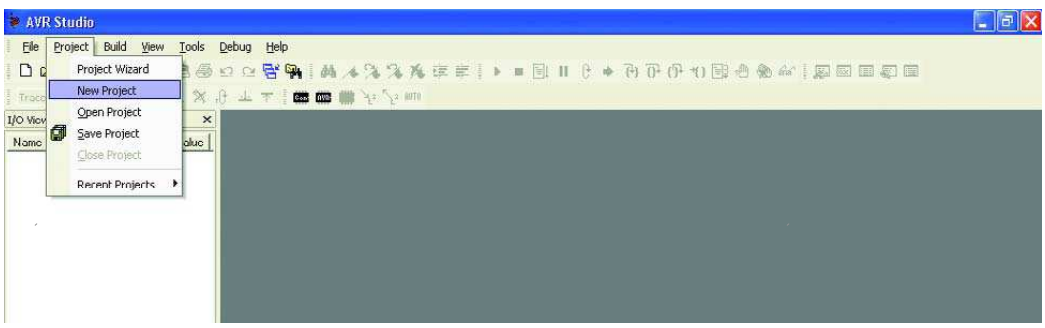
3.2.7 Меню Debug

Это меню позволяет получить доступ к инструментам отладки. При программировании набора MicroCamp эти инструменты практически не нужны.

3.3 Создание проекта на языке C в AVR Studio

3.3.1 Запустите AVR Studio. Если выполняется какой-либо проект, его необходимо закрыть Project → Close Project.

3.3.2 Для создания нового проекта выбирайте Project → New Project.



3.3.3 Появится окно свойств проекта. Установите следующие параметры:

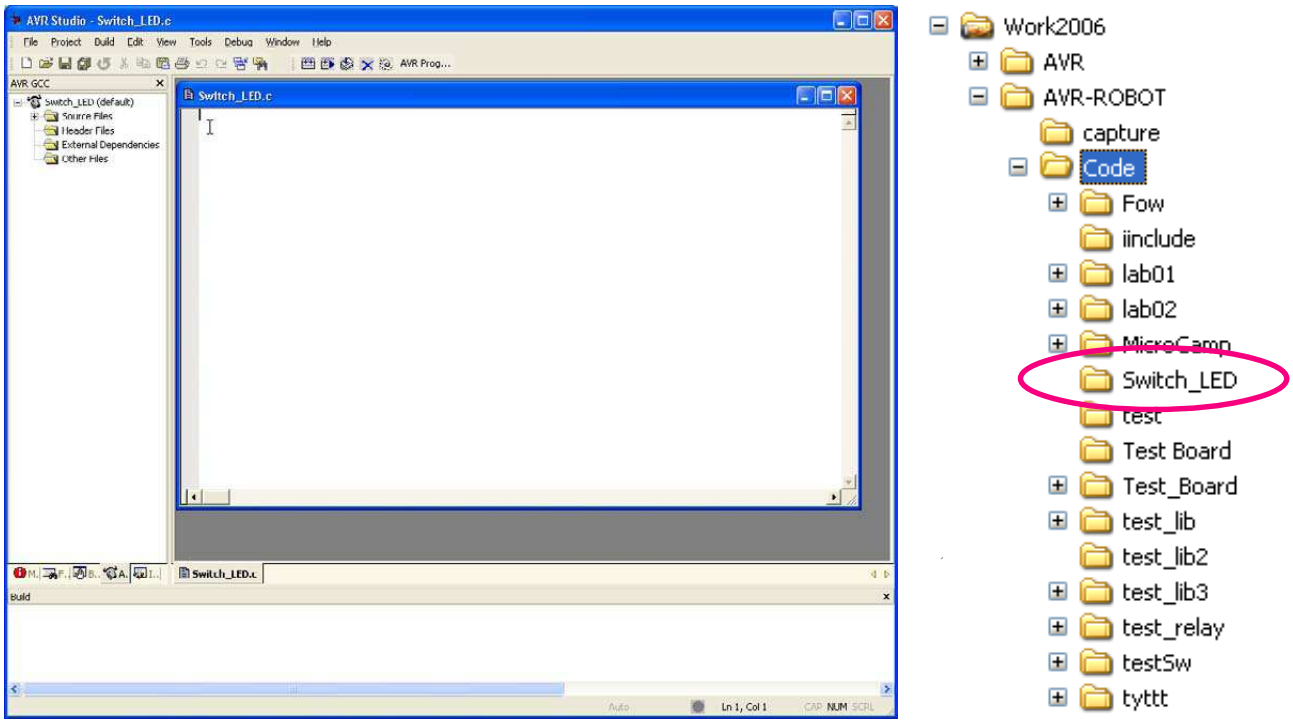
3.3.3.1 В окне Project type выделите строку AVR GCC: для выбора проекта на языке C.

3.3.3.2 Задайте имя проекта Switch_LED (пример). Будет создана начальная часть файла проекта. В этом проекте файл главного модуля будет иметь имя Switch_LED.c.

3.3.3.3 Задайте адрес проекта в Location:

Пример: G:\Work2006\AVR-ROBOT\Code. После этого нажмите кнопку Finish.

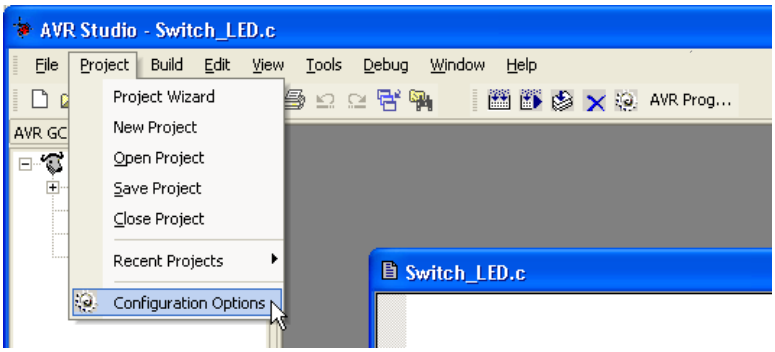
3.3.4 Рабочая область проекта Switch_LED будет выглядеть как на рисунке ниже.



Папка Switch_LED будет создана по адресу G:\Work2006\AVR-ROBOT\Code. В этой папке будут созданы файлы Switch_LED.aps и Switch_LED.c.

3.3.5 Далее необходимо определить параметры микроконтроллера и задать путь ко всем библиотекам и файлам, которые будут исполрзованы в проекте.

3.3.5.1 Выберите пункт Project → Configuration Options

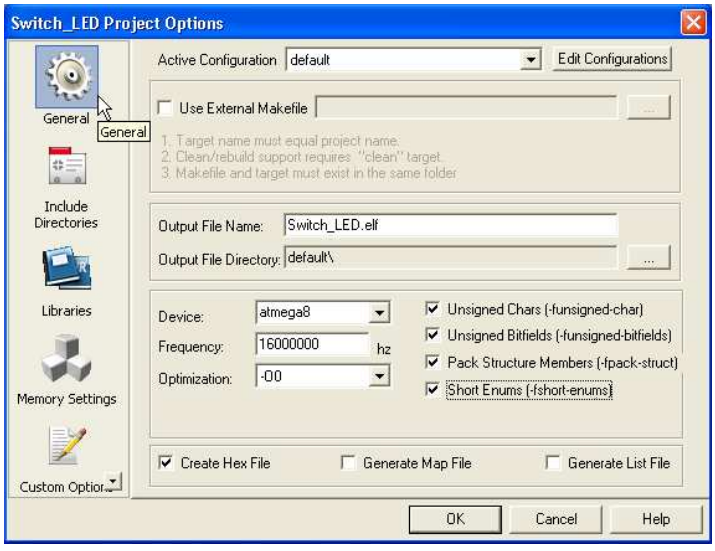


После этого появится окно Switch_LED Project Options. В левой части окна расположены 5 иконок:

- General
- Include Directories
- Libraries
- Memory Settings
- Custom Options

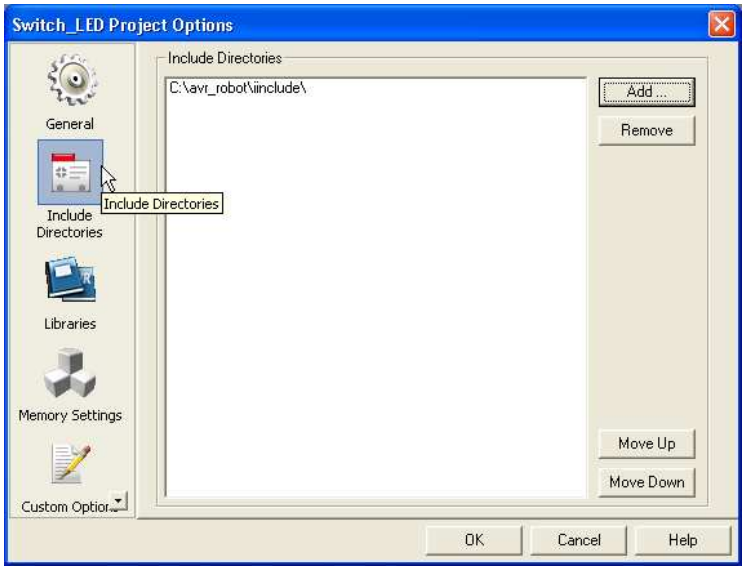
3.3.5.2 В пункте General задаются следующие параметры :

- Device : atmega8
- Frequency: 16000000 Hz

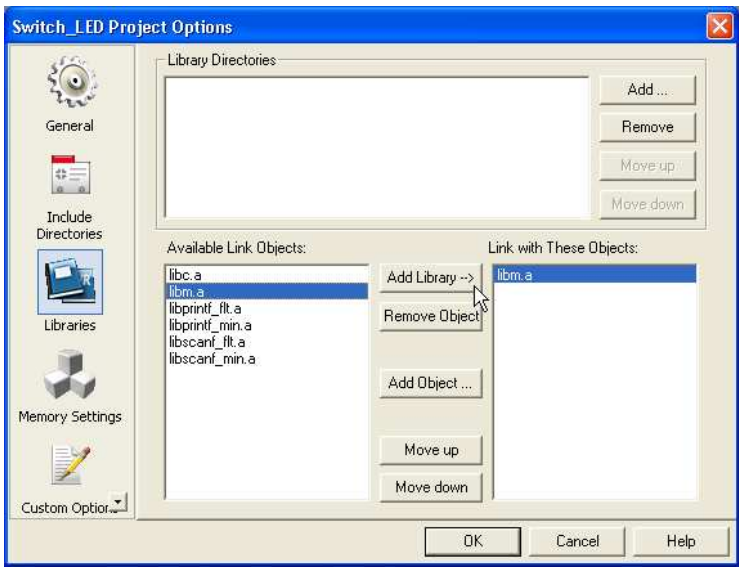


3.3.5.3 Иконка Include Directories предназначена для задания пути к библиотеке. Найдите и выделите файл библиотеки, затем нажмите кнопку Add. Например C:\AVR_ROBOT\include.

3.3.5.4 Иконка Libraries предназначена для связи всех библиотек с проектом.



3.3.5.5 В окошке Available Link Objects: выделите строку libm.a и нажмите кнопку Add Library. Нажмите кнопку OK для завершения.



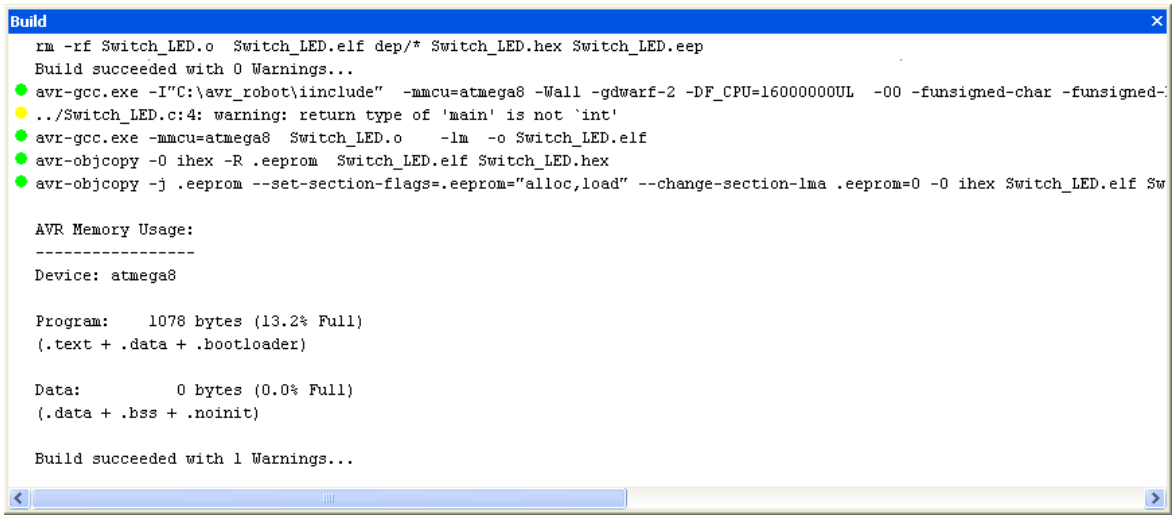
3.3.6 Затем введите код на языке C в файл Switch_LED.c. Этот код управляет микроконтроллером таким образом, что происходит включение и выключение светодиода при нажатии кнопки. Пример показан на листинге 3-1.

```
#include   <in_out.h>
#include   <sleep.h>
void  main()
{
  while(1)
  {
      if (in_d(2)==0)
      {
         toggle_c(5);
      }
      if (in_d(3)==0)
      {
         toggle_d(1);
      }
      sleep(200);
  }
}
```

Листинг 3-1 Код C файла Switch_LED.c

3.3.7 Компилируйте программу в файл Switch_LED.hex выбирая команду Build → Build или нажмите кнопку F7 или нажмите иконку 

Статус операции будет отображаться в окне внизу главного окна AVR Studio как показано на рисунке.



При возникновении любой ошибки, такой, как неправильная команда или ошибка связи, появляется окно Build Output. В этом случае необходимо отредактировать программу, устранить все ошибки и повторить компиляцию.

После компиляции будет создан файл Switch_LED.hex в директории проекта. Например: результирующий файл Switch_LED.hex хранится в директории G:\Work2006\AVRROBOT\Code\Switch_LED\ default.

3.4 Как редактировать существующий проект

Можно редактировать ранее созданный проект. Войдите в меню Project → Open Project и укажите путь к файлу проекта. Файл проекта имеет расширение .aps

Пример: Если необходимо редактировать проект Switch_LED, выбирайте Project → Open Project и укажите путь к файлу Switch_LED.aps. Откройте файл для редактирования. После редактирования можно сохранить файл с другим именем или с прежним.

3.5 Тестирование программы в микроконтроллере

После компиляции необходимо загрузить HEX-файл в микроконтроллер. В нашем случае результирующий файл будет Switch_LED.hex. Загрузка файла в микроконтроллер производится в следующем порядке:

3.5.1 Включите питание платы (выключатель POWER). Должен загореться зеленый светодиод.

3.5.2 Подключите ISP-кабель к программатору PX-400 и к плате MicroCamp.

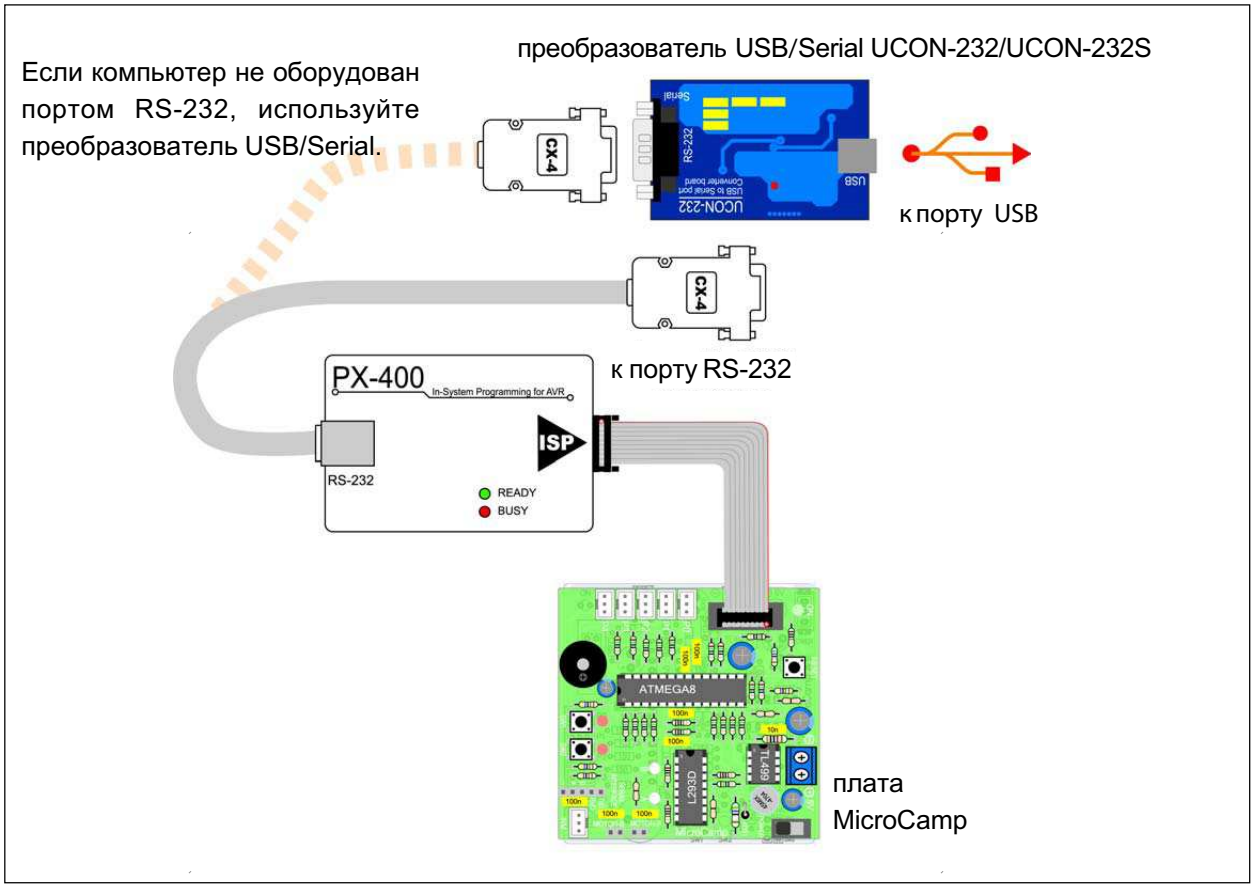
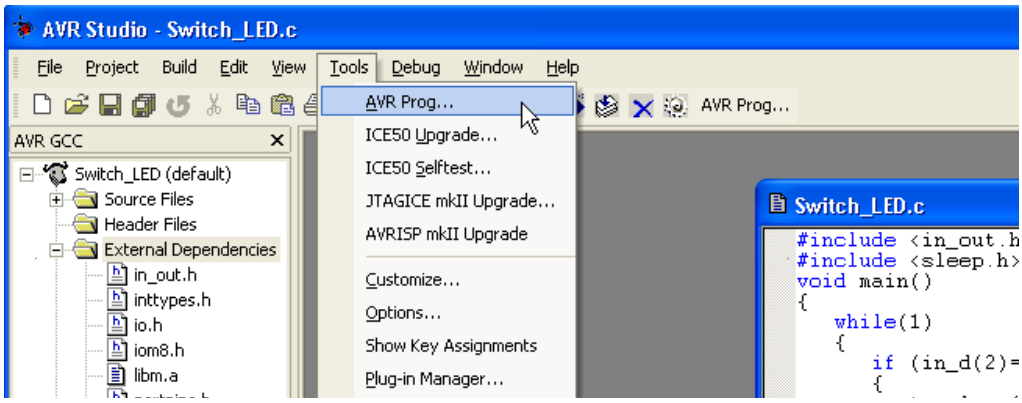


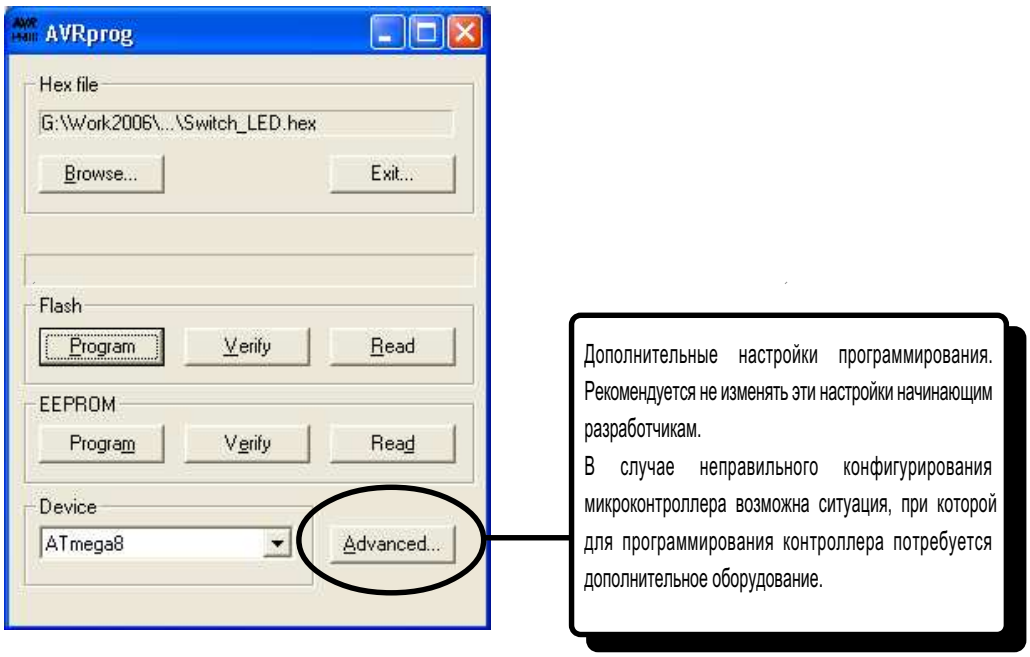
Рисунок 3-1 Схема подключений программатора PX- 400 к плате MicroCamp для загрузки программы

3.5.3 Запустите программу AVR Studio, выбирайте пункт меню Tool → AVR Prog...



3.5.4 Должно появиться окно AVRprog.

3.5.5 В окне AVRprog нажмите кнопку Browse для задания пути к файлу Switch_LED.hex или любому hex-файлу, предназначенному для загрузки.



3.5.6 Нажмите кнопку Program в группе Flash. Файл Switch_LED.hex будет загружен в микроконтроллер ATmega8 на плате MicroCamp.

3.5.7 Когда загрузка заканчивается, программа запускается автоматически. Нажимайте кнопки SW1 и SW2 на плате MicroCamp. Обратите внимание на состояние индикаторов LED.

Индикатор LED будет включаться и выключаться при нажатии на кнопки и мигать когда кнопки не нажаты.

Глава 4
Библиотека функций на языке C

В языке C функция соответствует подпрограмме или процедуре. Функция предоставляет удобный путь для инкапсуляции некоторых вычислений, которые могут быть использованы впоследствии. Правильно разработанные функции позволяют значительно упростить дальнейшее проектирование.

Все программы на языке C должны иметь «главную» функцию, которая содержит код, исполняемый в первую очередь. Остальные подпрограммы на языке C подключаются к главной функции. Таким образом, механизм поддержки функций является основным в программировании на языке C.

4.1 Объявление функций

Формат объявления:

```
return_type  function_name(parameter1,parameter2,
...) {
command_list 1;
.....................
.....................
command_list n;
}
```

function_name - имя функции

return_type - тип данных, возвращаемых функцией. В этой функции команда return(value) используется для пересылки результирующих данных. Типы принимаемых и пересылаемых данных должны совпадать. Функции без возвращаемых данных должны содержать void в return_type.

Параметр - это данные, передаваемые в функцию. Некоторые функции требуют много параметров, некоторые функции работают без параметров. Если параметры не требуются, необходимо указывать void.

command_list 1...command_list n - тело функции, содержащее все необходимые команды. Команды разделяются точкой с запятой.

4.2 Как использовать функции

Все функции, объявленные в программе на языке С могут быть вызваны в "главной" функции. При вызове функции разработчик должен указать имя функции и все параметры, которые необходимы данной функции. Данные, которые передаются всем параметрам каждой функции называются “Аргумент”.

Вызов функции производится следующим образом:

```
function_name (agument1, agument2,...)
```

function_name- имя вызываемой функции.

agument - данные, передаваемые от параметров функции. Если у функции нет параметров, то аргументы также не требуются.

Пример 4-1

```
void tone (void)
{ sound (3000,100);   // функция генератора звука;
                      // генерирует сигнал 3 кГц на 0.1 секунды
sleep (1000);         // задержка на 1 секунду
sound (3000,100);     // функция генератора звука
                      // генерирует сигнал 3 кГц на 0.1 секунды
}
```

Выше показан пример объявления функции tone. Эта функция не возвращает результатов и не требует параметров. Эта функция генерирует сигнал с частотой 3 кГц с длительностью 0.1 секунды и повторяет его через 1 секунду.

Эту функцию можно использовать в главной следующим образом:

```
void main()
      {
      ..............        // любые инструкции
      tone() ;              // вызов функции tone
      ..............        // любые инструкции
      }
```

Важно: для работы этой функции необходимо подключить 2 библиотеки в программу; sound.h и sleep.h

Пример 4-2

```
      void tone(unsigned int delay)
{
      sound(3000,100);      // генерация звука;
      sleep(delay);         // задержка на заданную величину;
      sound(3000,100);      // генерация звука;
}
```

Этот пример отличается от предыдущего в функции Sleep. Для функции необходим параметр "delay", который объявлен в функции tone для задания времени задержки в миллисекундах.

4.3 Библиотека

Библиотека - это файл, содержащий одну или несколько схожих функций. Обычно, название библиотеки отражает назначение содержащихся в ней функций.

Для использования библиотеки разработчик должен ее объявить в заголовке главной программы на языке С. Для корректной работы библиотеки необходимо правильно задать путь к ее файлу при создании проекта AVR Studio.

4.3.1 Как создать библиотеку

Файл библиотеки подобен программе на языке С за исключением того, что в нем отсутствует главная функция. После написания кодов их необходимо сохранить как файл с расширением .h. Например, создадим библиотеку; led1.h.

Необходимо выполнить следующие шаги:

(1) Создать новый файл File → New File.

(2) Написать коды функции Blink:

```
   void sleep(unsigned int ms)
                              {
              unsigned int i,j;
              for(i=0;i<ms;i++)
            for(j=0;j<795;j++);
}
void blink(unsigned int cnt)
{
      unsigned int _cnt=0;
      DDRC |= _BV(5);       // установить PC5 как выход
      while(_cnt < (cnt*2)) // проверка счетчика
            {
                  PORTC ^= _BV(5);      // инверсия бита PC5
                  sleep(300);           // пауза 0.3 секунды
                  _cnt++;
            }
}
```

(3) Сохранить файл File → Save As... Необходимо сохранить как файл .h. Библиотека led1.h готова к использованию.

4.3.2 Использование библиотеки

После создания файла библиотеки разработчик может вызывать все функции, содержащиеся в ней, после подключения библиотеки в заголовке программы C.

```
#include    <library_filename>
```

или

```
#include    “library_filename”
```

Команда #include помогает программе C использовать функции, содержащиеся в библиотеке.

Пример 4-6

(1) Создайте новый проект; test_lib

(2) Введите следующий код C в окне test_lib.c

```
#include <in_out.h>          // стандартная библиотека
#include <led1.h>            // получение функции blink из библиотеки
void main()
{
    blink(10);               // мигнуть светодиодом LED 10 раз
}
```

Описание:

Программа test_lib использует 2 файла библиотек. Одна библиотека стандартная для конфигурирования портов микроконтроллера ATmega8 (in_out.h). Другая библиотека led1.h создана нами. Библиотека led1 содержит 2 функции ; blink и sleep(). Функция blink настраивает порт PC5 как выход для управления светодиодом LED и формирует на выходе логические "1" и "0". Функция sleep() определяет паузы при управлении светодиодом LED. Функция blink отрабатывет заданное число раз .

(3) Задайте путь к файлу библиотеки led1.h в меню Project → Configuration Options

(4) Скомпилируйте проект. Должен быть создан файл test_lib.hex.

(5) Загрузите файл test_lib. hex в микроконтроллер .

(6) Наблюдайте результат.

Светодиод LED на выводе PC5 микроконтроллера ATmega8 мигнет 10 раз.

4.4 Типы данных при программировании на C

При программировании на языке C в WinAVR могут быть использованы следующие типы данных:

Тип данных	Размер
`char`	8-бит целочисленный со знаком. От -128 до +127.
`unsigned char`	8-бит целочисленный без знака. От 0 до +255.
`int`	16-бит целочисленный со знаком. От -32,768 до 32767.
`unsigned int`	16-бит целочисленный без знака. От 0 до +65535.
`long`	32-бит целочисленный со знаком. От -2,147,483,648 до +2,147,483,647.
`unsigned long`	32-бит целочисленный без знака. От 0 до +4294967295.
`long long`	64-бит целочисленный со знаком. От -9223372036854775808 до + 9223372036854775807.
`unsigned long long`	64-бит целочисленный без знака. От 0 до +18446744073709551616.
`float and double`	32-бит с плавающей запятой.
`arrays`	Группа данных или переменных одного типа, хранящихся в смежных адресах.
`pointers`	Указатели для задания адреса.
`structures`	Группа данных или переменных разных типов.

4.5 Системы счисления в языке C

Компилятор WinAVR поддерживает 3 системы счисления в программах C.

1. Десятичные числа
2. Двоичные числа. Формат 0bBBBBBBBB. Здесь, B это 0 или 1
3. Шестнадцатиричные числа. Формат 0xHHHHHHHH. Здесь, H принимает значения 0...9, A...F.

Пример 4-7

Двоичное восьмиразрядное число 0b10010010 соответствует 146 в десятичной системе .

Вычисляется: $(1x2^7) + (0x2^6) + (0x2^5) + (1x2^4) + (0x2^3) + (0x2^2) + (1x2^1) + (0x2^0) = 146_{10}$

Пример 4-8

Двоичное число 16-бит; 0b1111010011101101 соответствует 62701 в десятичной системе.

Вычисляется: $(1x2^{15}) + (1x2^{14}) + (1x2^{13}) + (1x2^{12}) + (0x2^{11}) + (1x2^{10}) + (0x2^9) + (0x2^8) + (1x2^7) + (1x2^6) + (1x2^5) + (0x2^4) + (1x2^3) + (1x2^2) + (0x2^1) + (1x2^0) = 62701_{10}$

Пример 4-9

Шестнадцатиразрядное число; 0xFF соответствует 255 в десятичной системе.

Вычисляется: $(15x16^1) + (15*16^0) = 255_{10}$ и 0xFF → 0b11111111 в двоичной системе.

Пример 4-10

Шестнадцатиразрядное число; 0x31 соответствует 49 в десятичной системе.

Вычисляется: $(3x16^1) + (1x16^0) = 49_{10}$ и 0x31 → 0b00111111 в двоичной системе.

4.6 Объявление переменных

Объявление переменных в программах на C WinAVR соответствует программированию на ANSI-C. Формат объявления:

```
type variable_name;
```

Здесь

`type` тип данных результата `variable_name` имя

переменной :

```
int a;          // объявление целочисленной переменной a
long result;    // объявление переменной result как тип long
float start;    // объявление переменной start как тип float
int x,y;        // объявление 2-целочисленных переменных: x и y.
float p,q,r;    // объявление 3 переменных: p, q и r. Тип данных float.
```

Также при объявлении переменной можно задавать ее начальное значение:

```
int x=100;                  // Объявление переменной x .
                            // Переменная типа int с начальным значением 100.
int x=15,y=78;              // Объявление переменных x и y.
                            // Тип данных integer и начальные значения
                            // x=15 и y=78.
long p=47L,q=31L;           // Объявление переменных p и q. Тип данных long.
                            // и начальные значения p=47 и q=31.
```

4.7 Преобразование типов данных

Формат преобразования

```
(type)variable
```

Здесь type - желаемый тип данных результата

variable - переменная, которой необходимо преобразовать тип данных

Пример 4-11

```
int x=100;                  // Объявление переменной x как integer и присвоение значения 100.
float y=43.67,z;            // Объявление переменных y и z как float и присвоение y = 43.67.
z = y+(float)x ;            // Присвоение z = y + x.
                            // Переменная x исходно имеет тип int.
                            // Ее необходимо преобразовать во float командой (float)x.
                            // Результат z = 143.67.
```

Пример 4-12

```
int a=50;                   // Объявление переменной a как integer и задание
                            // начального значения 50.
long b=23L,c;               // Объявление переменных b и c как long и задание b = 23.
c = b*(long)a;              // Присвоение c = b * a.
                            // Переменная a исходно int. Ее тип отличается от b и c.
                            // Ее необходимо конвертировать в тип long командой.
                            //  (long)a
                            // Результат c = 1150.
```

4.8 Типы переменных компилятора WinAVR

4.8.1 Массивы

4.8.1.1 Одномерные массивы

Одномерный массив объявляется следующим

образом : `type name[size];`

Здесь:

`type` - тип данных элементов массива

`name` -имя массива

`size` - размер массива (опция)

Доступ к элементам массива производится следующим образом :

`name[index]`

Здесь `index` - адрес элемента массива. Этот параметр может быть числом или переменной, но должен обязательно быть целочисленным.

Пример 4-13

Объявление :

```
char arr[4];
```

Это значит, что `arr` массив переменных содержит 4 элемента :

`arr[0]` : первый элемент массива с индексом 0

`arr[1]` : второй элемент массива с индексом 1

`arr[2]` : третий элемент массива с индексом 2

`arr[3]` : четвертый элемент массива с индексом 3

`arr[0]`, `arr[1]`, `arr[2]` и `arr[3]` - переменные типа char. Размер каждой переменной 1 байт. Таким образом, для объявления массива переменных `arr` требуется 4 байта памяти.

Пример 4-14

```
char dat[8] = {1,3,5,7,9,11,13,15} ;
```

Это объявление массива dat. В нем 8 элементов, которым присвоены значения:

```
dat[0] = 1;
dat[1] = 3;
dat[2] = 5;
dat[3] = 7;
dat[4] = 9;
dat[5] = 11;
dat[6] = 13;
dat[7] = 15;
```

Доступ к элементам массива может производиться следующим образом:

```
char i , j ;
i = 3;
j = dat[i]; // j = dat[i] ==> j = dat[3] ==> j = 7
            /* Результат j = 7 */
```

Пример 4-15

```
char dat[4] = " abcd" ;
```

Это объявление массива dat. В нем 4 элемента, которым присвоены значения:

```
dat[0] = 'a';
dat[1] = 'b';
dat[2] = 'c';
dat[3] = 'd';
```

Доступ к элементам массива может производиться следующим образом:

```
char i , j      // j = dat[i] ==> j = dat[3] ==> j = 'd'
                /* Результат is  j = 'd' */
```

Массив может быть глобальным или локальным. Он может использоваться в качестве параметра для передачи данных в функцию.

4.8.1.2 Двумерные массивы

Двумерный массив объявляется следующим образом:

```
type  name[x][y];
```

Здесь:

`type` - тип данных переменных массива

`name` - имя массива

`x` - число строк массива

`y` - число столбцов массива

Пример:

```
int  a[2][5];
```

Это объявление двумерного массива “a”. Он содержит 10 целочисленных элементов.

```
a[0][0],   a[0][1],   a[0][2],   a[0][3],   a[0][4] ,
a[1][0],   a[1][1],   a[1][2],   a[1][3],   a[1][4]
```

Элементы массива назначаются следующим образом:

```
int  menu[3][4] ={{1,3,4,9} , {2,8,0,5}};
```

В результате получим :

```
menu[0][0] = 1     menu[0][1] = 3     menu[0][2] = 4     menu[0][3] = 9
menu[1][0] = 2     menu[1][1] = 8     menu[1][2] = 0     menu[1][3] = 5
menu[2][0] = 0     menu[2][1] = 0     menu[2][2] = 0     menu[2][3] = 0
```

Глава 5

Операторы компилятора WinAVR

В программах на языке С для компилятора Win AVR применяются три типа операторов: арифметические, логические и битовые.

5.1 Арифметические операторы

Эта группа операторов содержит в себе следующие:

Оператор	Значение
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления
++	Увеличение на 1
- -	Уменьшение на 1
+=	Прибавить к текущему значению переменной
- =	Вычесть из текущего значения переменной
*=	Умножить на текущее значение переменной
/=	Разделить на текущее значение переменной
%=	Остаток от деления текущего значения переменной

5.1.1 Сложение (+) и вычитание (-)

Пример 5-1

```
int  a = 12;
a = a + 3;
```

Результат: a = 15

Действие: вначале a = 12. К переменной a прибавляется 3 и результат сохраняется в a.

Пример 5-2

```
int  a = 12;
a = a - 3;
```

Результат: a = 9

Действие: вначале a = 12. Из переменной a вычитается 3, и результат сохраняется в переменной a.

5.1.2 Деление / и %

Разница в использовании этих операторов заключается в следующем:

1. / деление двух чисел с целочисленным результатом.
2. % деление двух чисел, результатом является целочисленный остаток от деления.

Пример 5-3

```
int  x , y , z;
x = 10;
y = x/3;
z = x%3;
```

Результат: y = 9 и z = 1

Действие:

`y = x/3;` → `y = 10/3` → `y = 3` (целочисленный результат)

`z = x%3;` → `z = 10%3` → `z = 1` (результат - остаток от деления)

5.1.3 Операторы ++ и - -

Пример 5-4

```
int  y = 5;
y++;
```

Результат: y = 6

Действие: вначале y = 5. Затем y+1 = 6 и сохраняется в y. Здесь команда y++; работает аналогично y = y + 1.

Пример 5-5

```
int  y =
5;
y - -;
```

Результат: y = 4

5.1.4 Операторы += и - =

Операторы работают следующим образом:

```
y+=a;      действует аналогично   y = y + a
y-=a;      действует аналогично   y = y - a;
```

Пример 5-6

```
int  x = 100;
x += 10;
```

Результат: x = 110

5.1.5 Операторы *= , /= и %=

Операторы работают следующим образом:

```
y *=a;      действует аналогично   y = y * a;
y /=a;      действует аналогично   y = y/a;
y %=a;      действует аналогично   y = y%a;
```

Пример 5-7

```
int  x, y, z;
x = y = z = 120;
x *= 4;
y /= 4;
z %= 4;
```

Результат: x = 480 , y = 30 и z = 0

5.2 Логические операторы

Результатом действия таких операторов является “1” в случае истинности и “0” в случае ложности условия. Операторы работают следующим образом:

Оператор	Значение
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
!	НЕ
&&	И
||	ИЛИ

Пример 5-8

a = 10, b = 4, c = 0xA0

Действие	Состояние	Результат
a>b	*true*	*1*
a>c	*false*	*0*
a>=c	*true*	*1 (*так как 0xA0 = 10*)*
a!= b	*true*	*1*
a!= c	*false*	*0*

5.2.1 Операторы ! , && и ||

Оператор ! (НЕ) действует следующим образом:

Действие	Результат
! false	true(1)
! true	false(0)

Результатом действия оператора НЕ является инверсия входного значения. Оператор &&(И) действует следующим образом:

Действие	Результат
false && false	false(0)
false && true	false(0)
true && false	false(0)
true && true	true(1)

Результатом И является false, если одно или оба условия ложны.

Оператор ||(ИЛИ) действует следующим образом:

Действие	Результат
`false || false`	`false(0)`
`false || true`	`true(1)`
`true || false`	`true(1)`
`true || true`	`true(1)`

Результатом действия оператора ИЛИ является true, если одно или оба условия истинны.

Пример 5-9

a = 10, b = 4 и c = 0xA0

Действие	Состояние	Результат
a>b	*true.*	*1*
a>b	*true*	*1*
a>c	*false*	*0*
a>=c	*true*	*1 (так как 0xA0 = 10)*
a != b	*true*	*1*
a != c	*false*	*0*

или

!(a>b)	*false*	*0*
!(a>c)	*true*	*1*
!(a>=c)	*false*	*0*
!(a != b)	*false*	*0*
!(a != c)	*true*	*1*

или

Действие	Результат
!(a>b) && (a>=c)	*false(0)*
(a != b) && (a>=c)	*true(1)*
(a != b) && !(a != b)	*false(0)*

и

!(a>b) || (a>=c)	*true(1)*
(a != b) || (a>=c)	*true(1)*
(a != b) || !(a != b)	*true(1)*
!(a>=c) || !(a != b)	*false(0)*

5.3 Битовые операторы

Операторы работают следующим образом:

Оператор	Значение
~	Инверсия бита
&	Битовое И
|	Битовое ИЛИ
^	Битовое исключающее ИЛИ
<<	Сдвиг влево
>>	Сдвиг вправо
<<=	Сдвиг влево и сохранение результата в переменной
>>=	Сдвиг вправо и сохранение результата в переменной
&=	Операция И и сохранение результата в переменной
|=	Операция ИЛИ и сохранение результата в переменной
^ =	Операция искл. ИЛИ и сохранение результата в переменной

5.3.1 Применение битовых операторов

Оператор ~ действует следующим образом:

Действие	Результат
~ 0	1
~ 1	0

Оператор & действует следующим образом:

Действие	Результат
0 & 0	0
0 & 1	0
1 & 0	0
1 & 1	1

Оператор | действует следующим образом:

Действие	Результат
0 | 0	0
0 | 1	1
1 | 0	1
1 | 1	1

Оператор ^ действует следующим образом:

Действие	Результат
0 ^ 0	0
0 ^ 1	1
1 ^ 0	1
1 ^ 1	0

Пример 5-10

Дано:

```
int  x,y,result1,result2,result3,result4;
x  = 0x9C;
y  = 0x46;
```

Найдите результат:

(1) `result1 = x&y;`

(2) `result2 = x|y;`

(3) `result3 = x^y;`

(4) `result4 = ~x;`

Решение:

Вначале преобразуем все значения в двоичный формат.

x = 0x9C → 0000000010011100 (потому, что разрядность 16-бит)

y = 0x46 → 0000000001000110

(1) result1 = (0000000010011100) & (0000000001000110)

```
0000000010011100
      AND
0000000001000110
0000000000000100    → 0x0004 или 0x04
```

(2) result2 = (0000000010011100) | (0000000001000110)

```
0000000010011100
      OR
0000000001000110
0000000011011110    → 0x00DE или 0xDE
```

(3) result3 = (0000000010011100) ^ (0000000001000110)

```
0000000010011100
      XOR
0000000001000110
0000000011011010    → 0x00DA или 0xDA
```

(4) result4 = ~(0000000010011100) Инверсия всех бит

1111111101100011 → 0xFF63

5.3.2 Операции битового сдвига

В таких операциях необходимо определять на какое число бит производится сдвиг:

```
dat = dat<<4;
```

Это значит, что переменная dat сдвигается влево на 4 бита, и результат сохраняется в переменной dat.

Другой пример:

```
dat = dat>>1;
```

В этом случае переменная dat сдвигается на один бит вправо, и результат сохраняется в переменной dat.

Пример 5-11

```
int dat, x1, x2;
dat = 0x93;
```

Определите результат:

```
(1) x1 = dat<<1;
(2) x2 = dat<<2;
```

Решение:

dat = 0x93 → 0000000010010011

```
dat   0000000010010011
X1    0000000100100110
X2    0000001001001100
```

(1) x1 равен сдвинутой влево на 1 бит переменной dat.

Здесь x1 = 0x0126 или 294 в десятичном формате.

(2) x2 равен сдвинутой влево на 2 бита переменной dat.

Здесь x2 = 0x024C или 588 в десятичном формате.

Пример 5-12

```
int  a , b , c;
a = 0x7A;
b = 0x16;
c = 0xFD;
```

Определите результат:

```
(1) a &= 0x3C;
(2) b |= 0x51;
(3) c ^= 0xD0;
```

Решение:

(1) Запись a &= 0x3C соответствует a = a & 0x3C. Необходимо вычислить 0x7A AND 0x3C и сохранить результат в a.
Это аналогично: a = (0000000001111010) & (0000000000111100)

0000000001111010 AND 0000000000111100

0000000000111000 → ***0x0038*** или ***0x38***

(2) Запись b |= 0x51 соответствует b = b |= 0x51. Необходимо вычислить 0x16 OR 0x51 и сохранить результат в b.
Это аналогично: b = (0000000000010110) |= (0000000001010001)

0000000000010110 OR 0000000001010001

0000000001010111 → ***0x0057*** или ***0x57***

(3) Запись c ^= 0xD0 соответствует c = c ^= 0xD0. Необходимо вычислить 0xFD XOR 0xD0 и сохранить результат в c.
Это аналогично: c = (0000000011111101) ^= (0000000011010000)

0000000011111101 XOR 0000000011010000

0000000000101101 → ***0x002D*** или ***0x2D***

Глава 6

Библиотеки и специализированные команды для набора MicroCamp

В состав набора MicroCamp входит большое количество библиотек для разработки и обучения. Среди них библиотека управления портами ввода/вывода, чтение данных с аналоговых входов, библиотека временных задержек, библиотека контроля моторов и воспроизведения звука.

Перечень библиотек комплекта:

- in_out.h Библиотека работы с цифровыми портами.
- sleep.h Библиотека временных задержек
- analog.h Библиотека чтения аналоговых данных с портов P0...P4
- led.h Библиотека управления светодиодом LED
- motor.h Библиотека управления мотором постоянного тока
- sound.h Библиотека генерации звука
- timer.h Библиотека управления таймерами

Все необходимые библиотеки должны храниться в папке проекта или путь к ним должен быть правильно указан в проекте. Все дополнительные библиотеки хранятся в папке MicroCamp_include на компакт-диске из комплекта.

6.1 Команды в библиотеке in_out.h

6.1.1 Функция чтения цифровых портов

in_b : функция чтения порта B

in_c : функция чтения порта C

in_d : функция чтения порта D

Формат функции:

```
char in_a(x)
char in_b(x)
char in_c(x)
char in_d(x)
```

Параметр:

x - определяет номер используемого порта. Значение от 0 до 7.

Возвращаемое значение:

“0” или “1”

Пример 6-1

```
char x=0;                    // Объявление переменной x
x = in_b(2);                 // Чтение значения с порта PB2 и запись в x
```

Пример 6-2

```
char x=0;                    // Объявление переменной x
x = in_d(4);                 // Чтение значения с порта PD4 и запись в x
```

Пример 6-3

```
#include <avr/io.h>          // Подключение стандартной библиотеки работы с портами
#include <in_out.h>          // Подключение библиотеки контроля портов
#include <sound.h>           // Подключение библиотеки формирования звука
void main()
{
      while(1)
      {
            if (in_d(2)==0)                 // Проверка нажатия кнопки SW1
            {
                  sound(3000,100);          // Формирование звука при нажатии SW1
            }
      }
}
```

6.1.2 Функция записи данных в порт

Эта функция определяет порт, конфигурирует его на выход и посылает в него данные. Эта функция не возвращает никаких параметров.

`out_b` : посылка данных в порт B

`out_c` : посылка данных в порт C

`out_d` : посылка данных в порт D

Формат функции:

```
out_b(char _bit,char _dat)
out_c(char _bit,char _dat)
out_d(char _bit,char _dat)
```

Параметр:

`_bit` - выбор номера порта. Значения от 0 до 7.
`_dat` - задание выходного значения “0” или “1”.

Пример 6-4

```
out_c(5,0);                    // Выдача логического "0" в порт PC5
out_d(1,1);                    // Выдача логической "1" в порт PD1
```

Пример 6-5

```
#include <avr/io.h>            // Подключение стандартной библиотеки работы с портами
                               // Подключение библиотеки контроля портов
#include <sound.h>             // Подключение библиотеки формирования звука
void main()
{
    while(1)                   // Цикл
    {
        out_d(1,1);            // Выдача "1" в порт PD1. Включается индикатор LED2.
        sleep(300);            // Задержка 0.3 секунды.
        out_d(1,0);            // Выдача "0" в порт PD1. Выключается индикатор LED2.
        sleep(300);            // Задержка 0.3 секунды.
    }
}
```

6.1.3 Функция инвертирования значения на выходе порта

`toggle_b` : инвертирование выхода порта B

`toggle_c` : инвертирование выхода порта C

`toggle_d` : инвертирование выхода порта D

Формат функции:

```
toggle_b(x)
toggle_c(x)
toggle_d(x)
```

Параметр:

x - определяет номер используемого порта. Значение от 0 до 7.

Пример 6-6

```
toggle_c(5);      // инвертирование значения на выходе порта PC5
toggle_d(1);      // инвертирование значения на выходе порта PD1
```

6.2 Функция задержки в библиотеке sleep.h

Эта библиотека содержит одну функцию. Это функция sleep, используется для задержки на заданное число миллисекунд.

Формат функции:

```
void sleep(unsigned int ms)
```

Параметр:

время задержки в миллисекундах. Значения от 0 до 65,535.

Пример 6-7

```
sleep(20);        // задержка на 20 миллисекунд
sleep(1000);      // задержка на 1 секунду
```

6.3 Библиотека analog.h: чтение аналоговых входов

6.3.1 Аналоговые функции

Это функции считывания аналоговых значений. Чтение производится с портов PC0...PC4. Аналоговые сигналы проходят с аналоговых входов на АЦП внутри микроконтроллера ATmega8. Разрешение преобразователя 10-бит. Выходные значения лежат в диапазоне от 0 до 1,023 в соответствии с напряжением на входе от 0 до 5В.

Формат функции:

```
unsigned int analog(unsigned char channel)
```

Параметр:

`channel` - задает номер входа.
Значения от 0 до4. Соответствует PC0...PC4.

Возвращаемое значение:

Число после преобразователя АЦП, в диапазоне от 0 до 1,023 в десятичной системе.

Пример 6-8

```
int adc_val=0;          // определение переменной для хранения результата
adc_val = analog(0); // чтение из аналогового порта 0 (PC0) и запись в adc_val.
```

6.4 Функция мигания светодиода в библиотеке led.h

На плате MicroCamp установлены 2 светодиода на портах PC5 (LED1) и PD1(LED2). Мигание светодиода - простая функция, которая выводит в порт “0” и “1” поочередно. Эту функцию можно использовать совместно с другими функциями из библиотеки led.h.

`led1_on()` : включить мигание LED1 (PC5)

`led1_off()` : выключить мигание LED1 (PC5)

`led2_on()` : включить мигание LED2 (PD1)

`led2_off()` : выключить мигание LED2 (PD1)

Пример 6-9

```
void main()
{
   led1_on();
                          //    LED1 будет мигать даже при завершении основной программы
}
```

6.5 Библиотека контроля моторов motor.h

6.5.1 функции управления моторами

Эти функции используются для контроля моторов постоянного тока при помощи платы MicroCamp.

Формат функции:

```
void motor(char _channel,int _power)
```

Параметр:

_channel - выбор номера канала управления. На плате MicroCamp установлено 2 канала; 1 и 2.

_power - задает скорость и направление вращения мотора. Диапазон от -100 до 100.
Если значение положительно (1...100), мотор вращается вперед. Если значение отрицательно (-1...-100), мотор вращается в другую сторону. Если значение равно 0, мотор остановится, но вал не блокируется.

Пример 6-10

```
motor(1,60);     // Вращение мотора 1 со скоростью 60%
.................
motor(1,-60);    // Вращение мотора 1 со скоростью 60% в обратную сторону
```

Пример 6-11

```
motor(2,100);    // Вращение мотора 2 с полной скоростью (100%)
```

6.5.2 Функция motor_stop

Останавливается двигатель и блокируется его вал.

Формат функции:

```
void motor_stop(char _channel)
```

Параметр:

_channel - выбор номера канала управления. Этот параметр может принимать 3 значения: 1- для остановки мотора OUT1, 2 - для остановки мотора OUT2, ALL - для отключения двух моторов.

Пример 6-11

```
motor_stop(1);        // остановка мотора 1
motor_stop(2);        // остановка мотора 2
motor_stop(ALL);      // остановка двух моторов (1 и 2).
```

6.5.3 Функция motor_off

Эта функция используется для остановки всех действий с моторами и отключения напряжения на всех выходах моторов. Эта функция аналогична заданию скорости вращения 0.

Формат функции:

```
void motor_off()
```

6.5.4 Функция forward

Эта функция используется для вращения мотора постоянного тока, соответствующего движению робота вперед.

Формат функции:

```
void forward(int speed)
```

Параметр:

`speed` - определяет скорость вращения мотора. Диапазон от 0 до 100.

6.5.5 Функция backward

Эта функция используется для вращения мотора постоянного тока, соответствующего движению робота назад.

Формат функции:

```
void backward(int speed)
```

Параметр:

`speed` - определяет скорость вращения мотора. Диапазон от 0 до 100.

6.5.6 Функция s_left

Эта функция предназначена для управления моторами таким образом, что производится поворот робота влево.

Формат функции:

```
void s_left(int speed)
```

Параметр:

`speed` - определяет скорость вращения мотора. Диапазон от 0 до 100.

6.5.7 Функция s_right

Эта функция предназначена для управления моторами таким образом, что производится поворот робота вправо.

Формат функции:

```
void s_right(int speed)
```

Параметр:

`speed` - определяет скорость вращения мотора. Диапазон от 0 до 100.

6.6 Библиотека формирования звука sound.h

Эта функция используется для задания частоты звукового сигнала, формируемого пьезо-излучателем на плате MicroCamp.

Формат функции:

```
void sound(int freq,int time)
```

Параметр:

`freq` - задает частоту в Гц.

`time` - задает время звучания в миллисекундах.

Пример 6-13

```
sound(2000,500); // формирует сигнал с частотой 2кГц, длительностью 0.5 секунды.
```

6.7 Функция подсчета времени в библиотеке timer.h

6.7.1 Функция timer_start

Запуск таймера. После запуска этой функции значение таймера будет сброшено.

Формат функции:

```
void timer_start(void)
```

6.7.2 Функция timer_stop

Эта функция останавливает таймер и обнуляет счетчик.

Формат функции:

```
void timer_stop(void)
```

6.7.3 Функция timer_pause

Эта функция ставит таймер на паузу. Текущее значение счетчика сохраняется.

Формат функции:

```
void timer_pause(void)
```

6.7.4 Функция timer_resume

Запускает счетчик после функции timer_pause.

Формат функции:

```
void timer_resume(void)
```

6.7.5 Функция msec

Считывание значения таймера в миллисекундах.

Формат функции:

```
unsigned long msec()
```

Возвращаемое значение:

Значение времени в миллисекундах. Тип данных “long”.

6.7.6 Функция sec

Считывание значения таймера в секундах.

Формат функции:

```
unsigned long sec()
```

Возвращаемое значение:

Значение времени в секундах . Тип данных “long”.

Пример 6-14

```
#include <in_out.h>
#include <sleep.h>
#include <timer.h>
void main()                  // Главная программа
{
   timer_start();            // Запуск таймера
   while(1)                  // Бесконечный цикл
   {
      if(msec()>500)         // Проверка значения таймера . Значение больше 500?
      {
         timer_stop();       // Остановка таймера и сброс значения
         toggle_c(5);        // Мигание индикатора каждые 0.5 сек
         timer_start();      // Запуск таймера
      }
   }
}
```

Глава 7

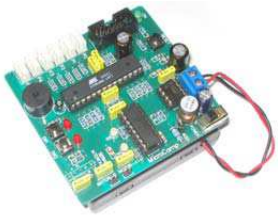
Построение робота из набора MicroCamp

Эта глава описывает практическое приложение микроконтроллерной платы MICROCAMP. При построении робота необходима интеграция знаний и технологий, которые включают электронику, программирование, механику и основы искусственного интелекта. Набор Microcamp позволяет изучать все эти области. В набор входят все необходимые детали для построения простого движущегося робота. Пользователь может изучать программирование и применение микроконтроллеров при построении робота и разработке соответствующего программного обеспечения

Робот на основе набора MICROCAMP содержит 2 мотора постоянного тока с редукторами для движения и 4 детектора для сбора информации. Среди них 2 детектора касания и 2 инфракрасных отражающих детектора для движения по линии.

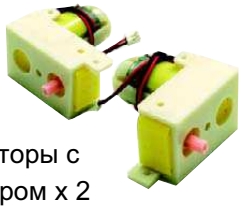
Список деталей

Круглая основа



Плата MicroCamp

Кожух x 1



48:1 моторы с редуктором x 2

Колеса с шинами x 2

Набор стоек x 1

Набор винтов и гаек x 1

Пластиковые крепления (прямые и угловые)

ИК - детекторы x 2

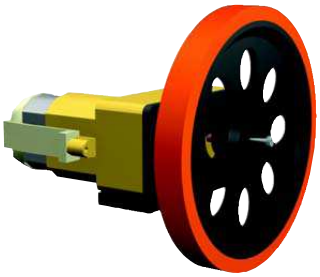
Контактные детекторы x 2

2мм шурупы x2

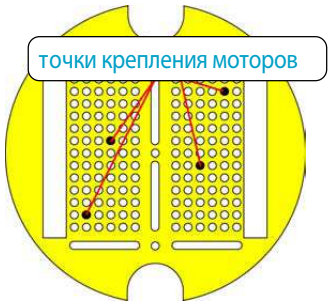
25мм металлические стойки x 2

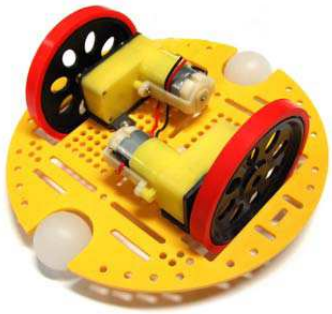
Сборка

1. Наденьте резиновые шины на колеса и наденьте их на оси моторов. Закрепите колеса при помощи шурупов 2мм.

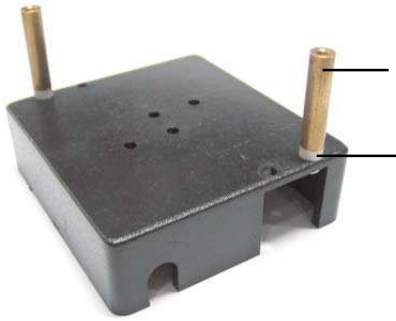


2. Установите моторы на круглом основании на позиции, указаные на рисунке ниже. Фиксируйте их при помощи четырех винтов 3 x 6мм.

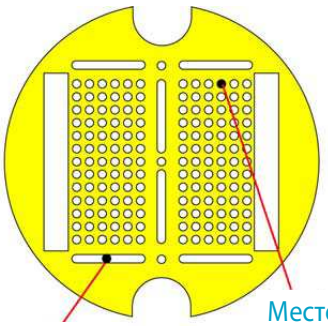


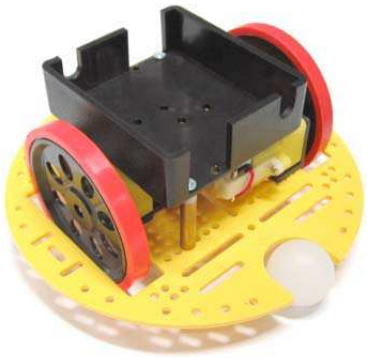


3. Вставьте 3 x 10мм через отверстия в углах кожуха и прикрутите металлические стойки 25мм через две пластиковые 3мм шайбы.

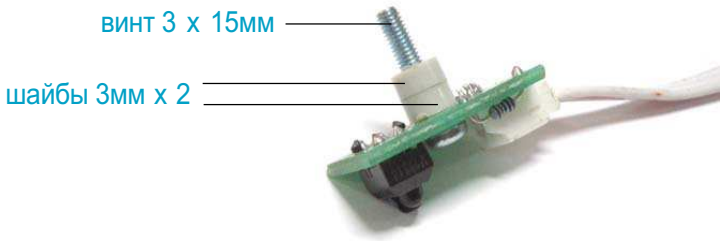


4. Установите кожух на круглое основание и зафиксируйте его винтами 3 x 10мм в указанных позициях.

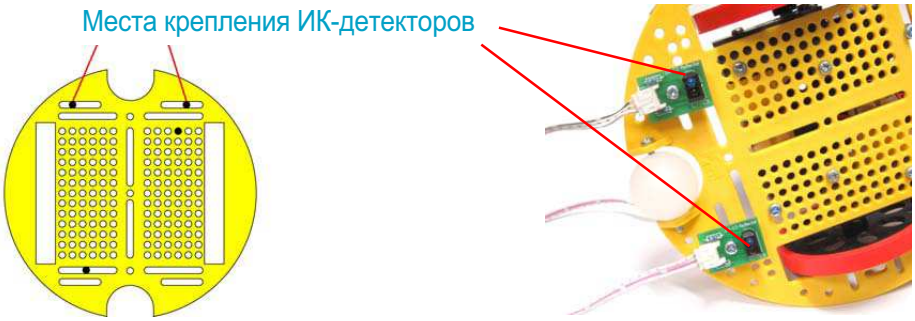




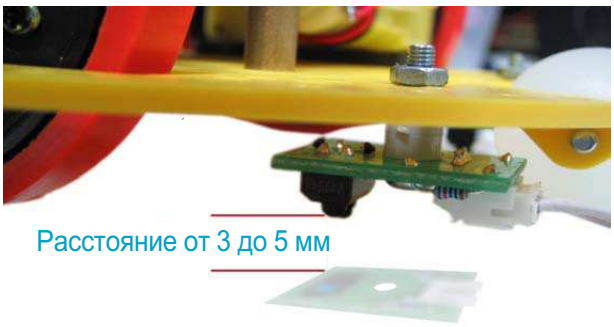
5. Вставьте винт 3x15мм в отверстие платы ИК-детектора, наденьте 2 шайбы 3мм. Также оформите второй детектор.

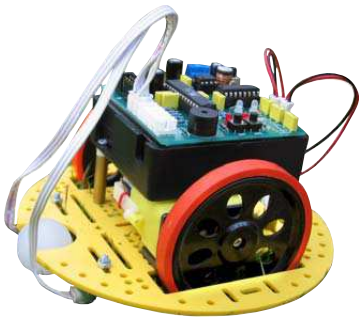


6. Присоедините оба детектора к круглому основанию через подходящие отверстия в передней части робота. Закрепите гайками 3мм.

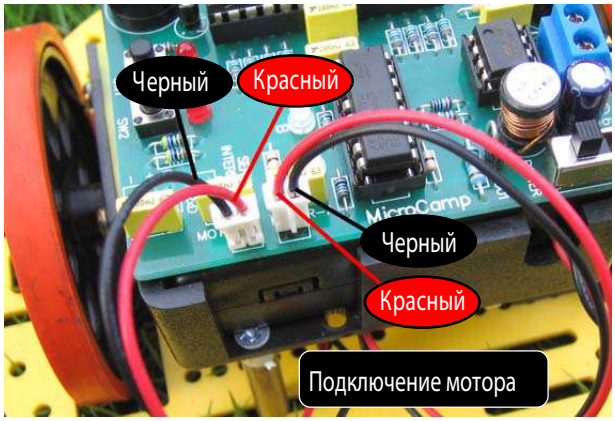


7. Проверьте расстояние от детекторов до пола. Расстояние должно составлять от 3 до 5 мм.

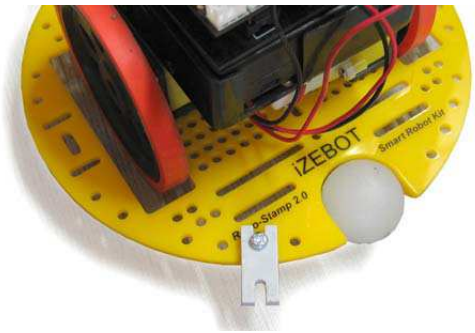




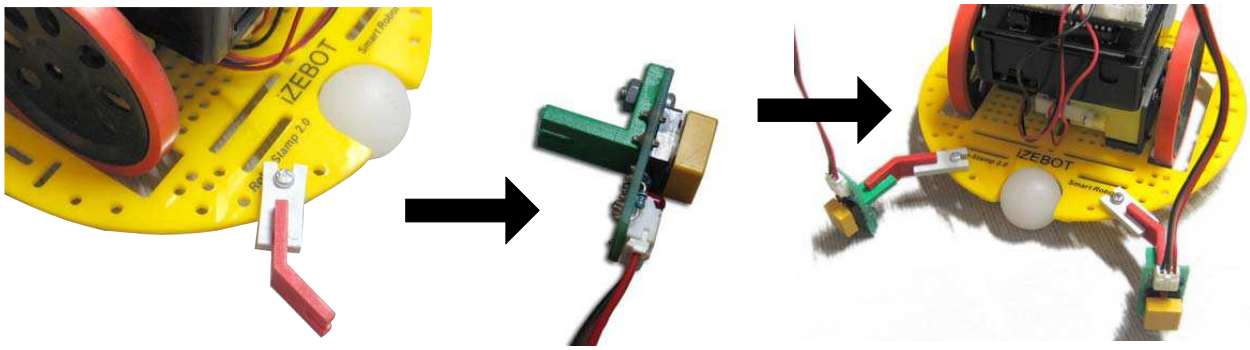
8. Вставьте плату MicroCamp в кожух. Подключите кабели детекторов и моторов как показано на рисунке (правый детектор к P0, левый детектор к P1).



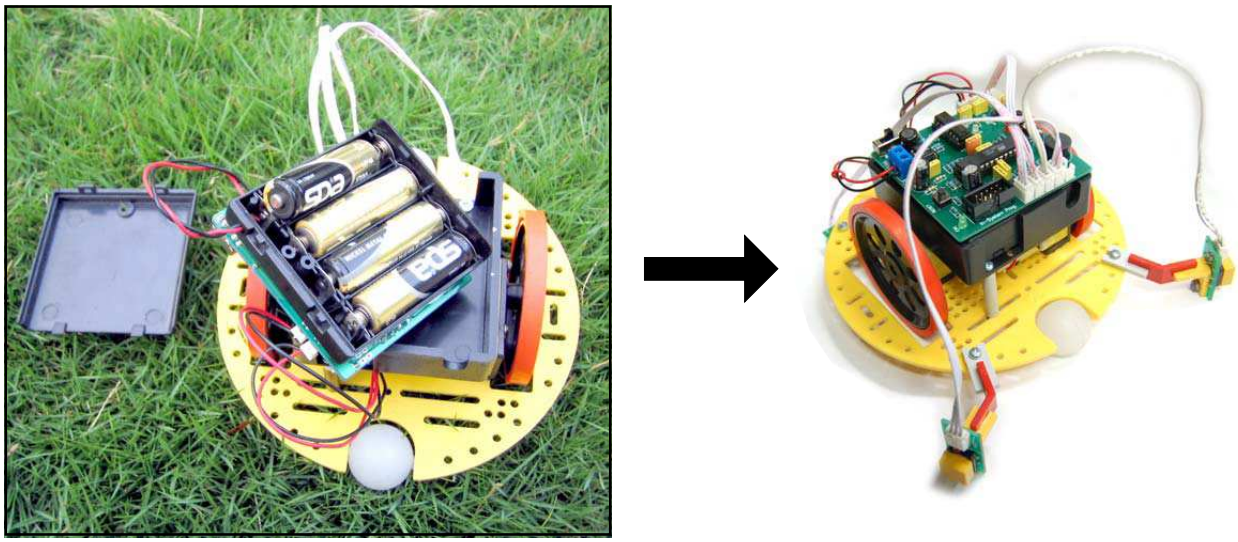
9. Присоедините прямое крепление к основанию робота на передней стороне справа винтом 3 x 10мм и гайкой 3мм. Аналогично прикрепите крепление слева.



10. Присоедините изогнутое крепление к концу прямого крепления. Присоедините угловое крепление к модулю контактного детектора винтом 3 x 10мм и гайкой 3мм. Присоедините получившуюся конструкцию к концу изогнутого крепления. Повторите то же с другой стороны.

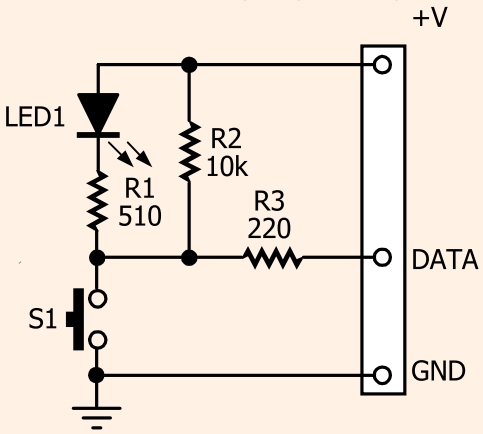


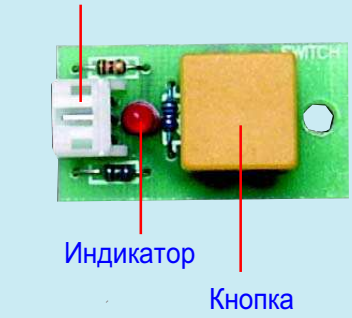
11. Подключите кабель левого контактного детектора к порту P2 (PC2), а правого детектора к порту P3 (PC3). Вставьте 4 AA батарейки в батарейный отсек с обратной стороны платы MicroCamp. Теперь робот MicroCamp готов к загрузке программы.



Принцип действия контактного детектора

Детектор MicroCamp имеет следующую схему:





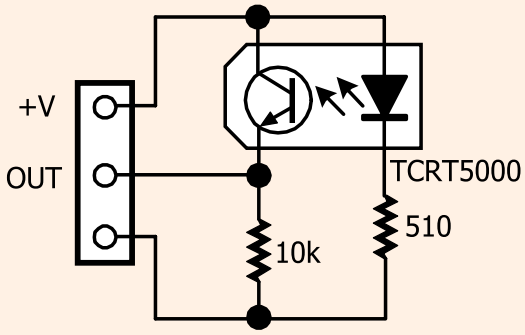
Детектор может находиться в двух состояниях :

Когда кнопка не нажата, DATA принимает значение "1"

Когда кнопка нажата, DATA принимает значение "0", и светодиод LED1 загорается.

Поскольку выходной сигнал принимает два состояния, детектор является цифровым.

Принцип действия ИК-детектора



Основой детектора является сенсор, который определяет отражение света в инфракрасном диапазоне. Детектор состоит из ИК-светодиода, который облучает поверхность и фото-транзистора, принимающего отраженный ИК-свет. Если нет принятого сигнала, на выходе OUT будет низкое напряжение. В случае приема ИК-света, через фототранзистор протекает маленький или большой ток, в зависимости от интенсивности принятого света, которая зависит от расстояния до отражающей поверхности. (Сенсор TCRT5000 может использоваться на расстояниях от 0.1 до 1.5 см).

На выходе OUT напряжение меняется от 0.5 до 5В, микроконтроллер MicroCamp измерит значение от 30 до 1023.

Задание 1
Базовое движение робота MicroCamp

Задание 1-1 Движение вперед и назад

A1.1 Откройте AVR Studio, создайте новый проект и введите программу C в соответствии с листингом A1-1. Скомпилируйте проект.

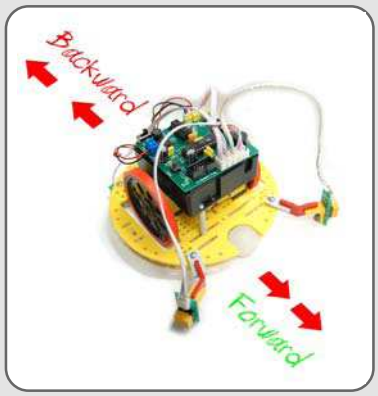
A1.2 Подключите программатор PX-400 к плате MicroCamp робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A1.3 Отключите питание и отсоедините ISP-кабель.

A1.4 Установите робота на ровную поверхность. Включите питание и наблюдайте за поведением робота. Робот MicroCamp начнет движение вперед. Оба индикатора моторов будут светиться зеленым светом. Через 1 секунду, оба индикатора изменят цвет на красный , и робот начнет движение назад.

Если поведение робота отличается от описанного выше, то необходимо пересоединить кабели моторов наоборот (поменять местами или изменить полярность). Делайте это до тех пор, пока робот не будет двигаться правильно. Робот будет продолжать движение вперед и назад до отключения питания.

```
#include <in_out.h>
#include <sleep.h>
#include <motor.h>  //      Библиотека контроля моторов
void main()
{
while(1)            //  Бесконечный цикл
{
forward(100);   //  Движение робота вперед
sleep(1000);    //  Пауза 1 секунда.
backward(100);  //  Движение робота назад
sleep(1000);    //  Пауза 1 секунда.
}
}
```



Листинг A1-1 Программа C для движения робота вперед и назад

Задание1-2 Движение по кругу

A1.5 Создайте новый проект и введите программу C в соответствии с листингом A1-2.

A1.6 Подключите программатор PX-400 к плате MicroCamp-робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A1.7 Отключите питание и отсоедините ISP-кабель.

A1.8 Установите робота на ровную поверхность. Включите питание и наблюдайте за поведением робота. Робот начнет движение после нажатия кнопки SW1 и будет двигаться по кругу, пока Вы не нажмете кнопку SW2.

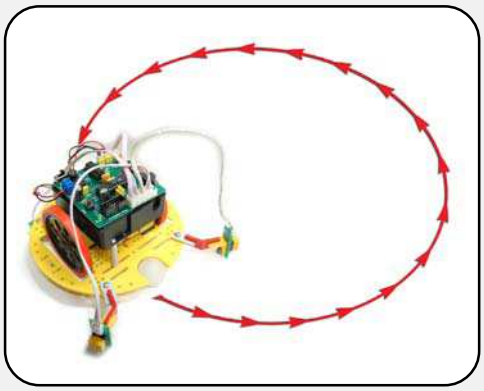
```
#include <in_out.h>
#include <sleep.h>
#include <motor.h>
void main()
{
   while(1)
   {
     while((in_d(2)==1));     // Контроль нажатия кнопки SW1
     motor(1,100);            // Включение мотора 1 на полную скорость
     motor(2,30);             // Включение мотора 2 на 30%
     while((in_d(3)==1));     // Контроль нажатия кнопки SW2
     motor_off();             // Выключение моторов
   }
}
```

Описание прграммы

В листинге A1-2, не используются команды «вперед» и «назад». Вместо этого используется функция MOTOR. При помощи этой функции можно независимо контролировать выходы драйверов моторов. Это значит, что можно независимо управлять скоростями моторов.

Если моторы крутятся с разной скоростью, то робот будет двигаться в направлении мотора с более низкой скоростью. Если разница скоростей значительна, то робот будет двигаться по кругу.

В этой программе использована команда While. Если кнопка SW1 в порту PD2 нажата, то логический “0” поступает в порт. После нажатия кнопки задаются параметры движения робота и программа переходит в ожидание нажатия кнопки SW2 в порту PD3. При нажатии на эту кнопку программа выключает оба мотора. Робот прекращает движение.



Листинг A1-2 Программа C для движения робота по кругу

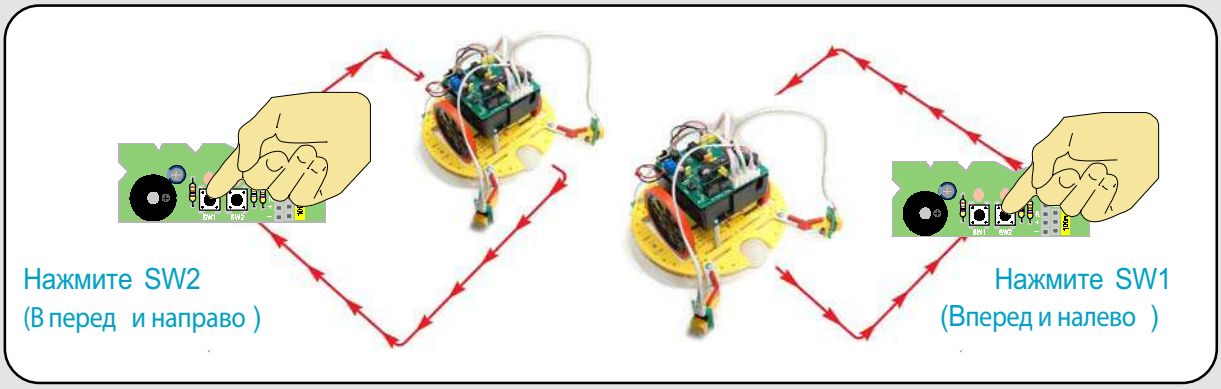
Задание 1-3 Движение по квадрату

A1.9 Создайте новый проект и введите программу C в соответствии с листингом A1-3. Подключите программатор PX-400 к плате MicroCamp - робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A1.10 Отключите питание и отсоедините ISP-кабель. Установите робота на ровную поверхность. Включите питание и наблюдайте за поведением робота.

Робот включается при нажатии SW1 или SW2. Если Вы нажмете SW1, робот будет двигаться вперед и вправо по квадрату. Нажатие SW2 приведет к движению вперед и влево.

```
#include <in_out.h>
#include <sleep.h>
#include <motor.h>
void main()
{
while(1)                          // Начало цикла
{
if (in_d(2)==0)                   // Проверка SW1
     {
        while(1)
        {
           forward(100);          // Вперед на полной скорости 0.9 секунды
           sleep(900);
           s_right(50);           // Вправо 50% скорости 0.3 секунды
           sleep(300);
        }
     }
     if (in_d(3)==0)              // Проверка SW2
     {
        while(1)
        {
           forward(100);          // Вперед на полной скорости 0.9 секунды
           sleep(900);
           s_left(50);            // Влево 50% скорости 0.3 секунды
           sleep(300);
        }
     }
   }
}
```



Листинг A1-3 Программа C управления движением робота

Задание 2

Контактное обнаружение объектов

Задание 2-1 Простое обнаружение столкновения

Это задание предназначено для изучения работы контактных детекторов, расположенных в передней части робота. После обнаружения столкновения, робот отъезжает назад и меняет направление движения.

A2.1 Создайте новый проект и введите программу C в соответствии с листингом A1-4. Скомпилируйте проект.

A2.2 Подключите программатор PX-400 к плате MicroCamp-робота. Включите робота.

A2.3 Загрузите HEX-файл в память микроконтроллера.

A2.4 Отключите питание и отсоедините ISP-кабель.

A2.5 Подготовьте место испытания робота с препятствиями.

A2.6 Включите питание и наблюдайте за поведением робота. Робот будет проверять состояние обоих контактных детекторов на портах PD2 и PC3. При прикосновении детектором к объекту в соответствующий порт поступит логический “0”.

При отсутствии препятствий робот движется вперед.

При срабатывании левого детектора, робот отъедет назад и затем повернет направо, чтобы объехать препятствие.

При срабатывании правого детектора, робот отъедет назад и затем повернет налево, чтобы объехать препятствие.

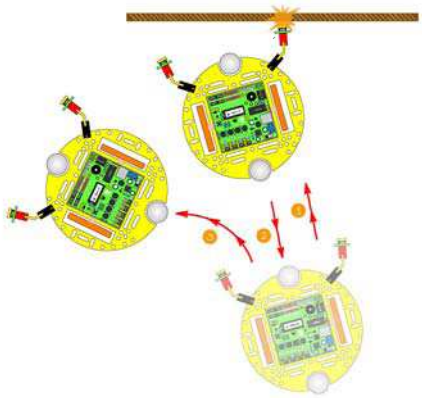
```
#include <in_out.h>
#include <sleep.h>
#include <motor.h>
void main()
{
while((in_d(2)==1));    // Проверка SW1 для запуска программы робота
while(1)                //  Начало цикла
{
if (in_c(2)==0)            // Проверка статуса правого детектора
     {
        backward(100);     //  Если есть препятствие, то робот движется назад 0.4 сек

        sleep(400);
        s_left(50);        //  затем разворачивается влево  0.3 сек
     sleep(300);
     }
     else if (in_c(3)==0) //  Проверка статуса правого детектора
     {
     backward(100);        //  Если есть препятствие, то робот движется назад 0.4 сек

        sleep(400);
        s_right(50);       //  затем разворачивается вправо  0.3 сек
     sleep(300);
     }
     else
     {
     forward(100);         //  Если нет препятствий, робот движется вперед

     }
   }
}
```

Поведение робота при препятствии слева



Поведение робота при препятствии справа

Листинг А2-1 Программа для контактного обнаружения препятствий

Задание 2-2 Выезд из угла

Когда робот попадает в угол, он не может оттуда выбраться, поскольку поочередно наталкивается то на левую, то на правую стены. Для решения этой проблемы необходимо модифицировать предыдущую программу (A2-1) так, как показано на A2-2.

A2.7 Создайте новый проект и введите программу C в соответствии с листингом A2-2.

A2.8 Подключите программатор PX-400 к плате MicroCamp-робота. Включите робота. Загрузите HEX-файл в память микроконтроллера. Отключите питание и отсоедините ISP-кабель.

A2.9 Подготовьте место испытания робота с препятствиями.

A2.10 Включите питание и наблюдайте за поведением робота.

Робот будет двигаться вперед и контролировать столкновения. Если это произойдет 5 раз подряд, то робот развернется на 180 градусов и будет двигаться в обратном направлении.

```
#include <in_out.h>
#include <sleep.h>
#include <motor.h>
#include <sound.h>              // Библиотека работы со звуком
void main()
{
   unsigned char cnt_=0;        // Объявление переменной для подсчета столкновений

   while((in_d(2)==1));         // Контроль SW1 для начала движения
     sound(3000,100);           // Звуковой сигнал
   while(1)                     // Цикл
   {
     if (in_c(2)==0)            // Проверка правого детектора
     {
        if ((cnt_%2)==0)        // Контроль счетчика столкновений
                                // Если счетчик не равен нулю, проверка предыдущего
                                // столкновения. Левое?
        {cnt_++;}               // Увеличение счетчика
        else                    // Если столкновение не левое, то
        {cnt_=0;}               // очистка счетчика
        backward(100);          // Движение назад 0.4 секунды
        sleep(400);             //
        s_left(50);             // Поворот влево
        if (cnt_>5)             // Проверка, что значение счетчика не превышает 5
        {
           sleep(700);          // Если превышает, поворот влево 0.7 секунды
        sound(3000,100);        // Вывод звука
        cnt_=0;                 // Сброс счетчика
        }
```

Листинг A2-2 Программа C для решения проблемы запирания робота в углу (продолжение на следующей странице)

```
        else                    // Если счетчик меньше 5,
        { sleep(300);}          // поворот 0.3 секунды
   }
     else if (in_c(3)==0)       // Контроль левого детектора
     {
        if ((cnt_%2)==1)        // Проверка: счетчик четный или нет
                                // Если да, то предыдущее столкновение правое.
        {cnt_++;}               // Счетчик увеличивается на 1
        else
        {cnt_=0;}               // Если нет,  то сброс счетчика
        backward(100);          // Движение назад 0.4 секунды
        sleep(400);             //
        s_right(50);            // Поворот направо 0.3 секунды
        sleep(300);             //
     }
     else                       // Если нет препятствий, то движение вперед
     {forward(100);}
   }
}
```

Листинг А2-2 Программа С для решения проблемы запирания робота в углу (окончание)

Глава 8

Работа с LCD-дисплеем

Дисплей SLCD16x2 это двухстрочный жидкокристаллический дисплей с 16 символами в строке и последовательным интерфейсом. Он получает данные по последовательному каналу и выводит их на дисплей. Скорость передачи данных составляет 2400 или 9600 bps, уровень задается 2 перемычками и может быть TTL или RS-232. Используется стандартный контроллер дисплея HITACHI HD44780 или SEIKO EPSON SED1278.

Обычно, дисплей требует не менее 6 проводов для подключения, но SLCD16x2 подключается 1 сигнальным проводом. Этот дисплей совместим с роботом MicroCamp.

8.1 ОписаниеSLCD16x2

8.1.1 Особенности

- Последовательный канал RS-232 или уровень TTL/CMOS.
- 1/8 или 1/16 заполнение задается перемычкой.
- Совместим с расширенным набором команд управления Scott Edwards's LCD Serial Backpack™ command.
- Простой интерфейс с микроконтроллером.
- Напряжение питания от +5 до 12В.

8.1.2 Настройка

На рис. 8-1 показана обратная сторона модуля SLCD16x2. Видно 4 перемычки для задания конфигурации.

(1) Задание режима команд: Модуль SLCD16x2 имеет 2 режима. Один стандартный (ST). Другой - расширенный набор команд (EX). Для работы с библиотеками MicroCamp установите стандартный режим (ST).

(2) Задание режима отображения: Заполнение 1/8 и 1/16. Заполнение 1/8 значит, что отображается 8 символов в строке. Заполнение 1/16 означает отображение 16 символов в строке. Обычно рекомендуется задавать 1/16.

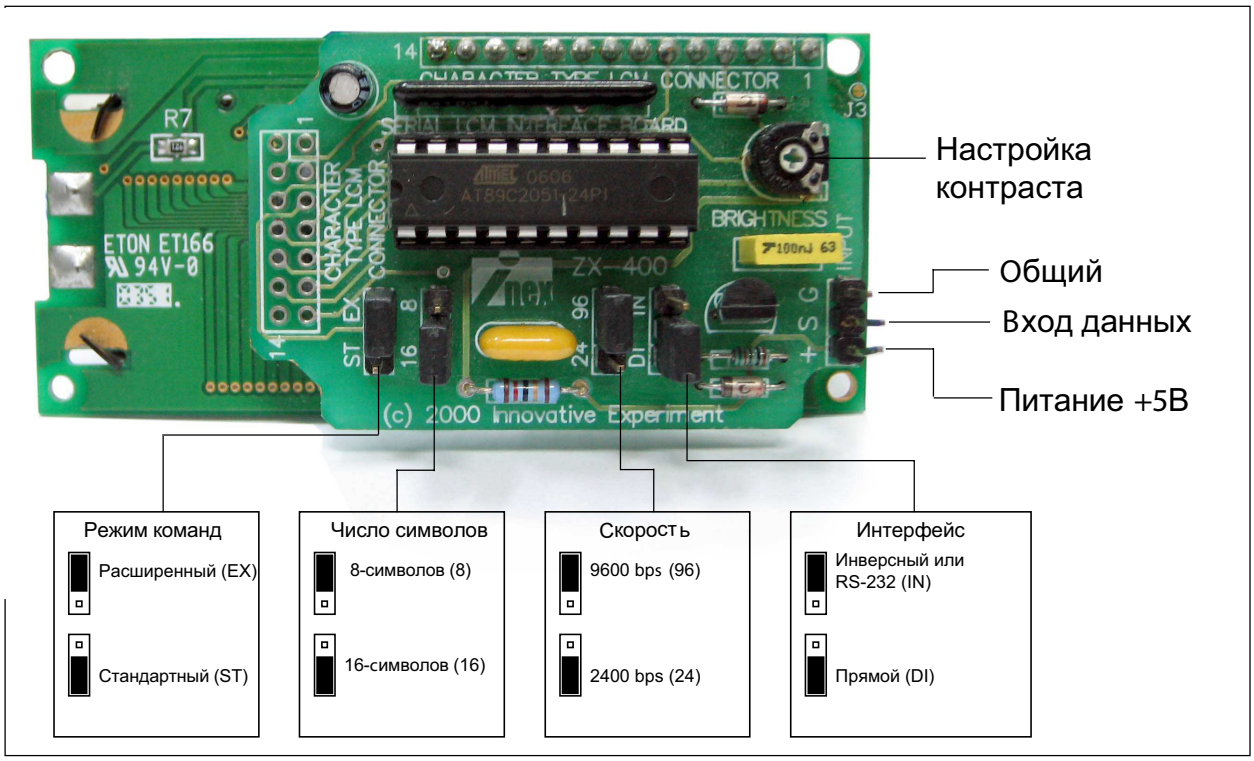


Рисунок 8-1 Установка перемычек в модуле SLCD16x2™

(3) Скорость передачи данных: 2400 или 9600 bps (бит в секунду), формат данных 8N1 (8-бит, без бита четности и 1 стоповый бит)

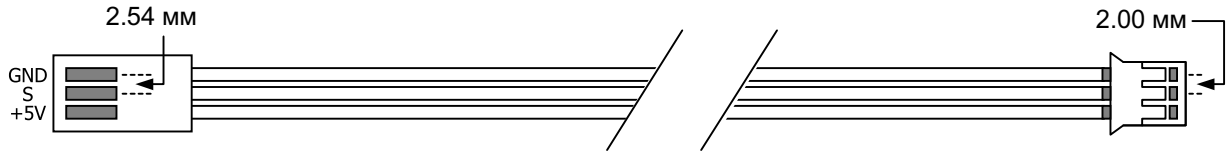
(4) Выбор интерфейса: Инвертированный TTL/CMOS или RS-232 (IN) и прямой TTL/CMOS (DI).

Потенциометр позволяет настроить контрастность дисплея SLCD16x2.

Сигнальный разъем имеет 3 контакта: Питание+5В (+), Вход данных (S) и Общий (G).

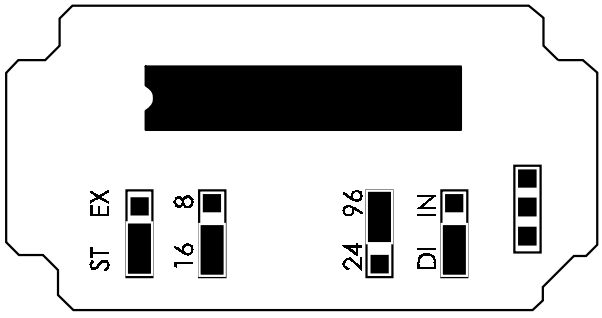
8.1.3 Подключение SLCD16x2 к плате MicroCamp

Необходимо использовать кабель JST3AB-8. Цоколевка этого кабеля показана ниже.



У кабеля JST3AB-8 с одной стороны контакты расположены через 2.54 мм (B-end), а с другой стороны через 2.00 мм (A-end). Сторона A-end подключается к разъему JST любого порта (P0...P4) платы MicroCamp. Сторона B-end подключается к разъему на модуле SLCD16x2.

После подключения, установите все перемычки как указано ниже.



- Режим команд Standard (ST).
- В строке 16-символов (16).
- Скорость 9600 bps (96).
- Интерфейс Direct (DI).

8.1.4 Посылка данных и команд в дисплей

После подключения и настройки модуля SLCD16x2 он может принимать последовательные данные. Для посылки данных необходимо выдать любое сообщение через последовательный порт, и оно отобразится на дисплее.

Для посылки команды необходимо использовать формат команд (см. рис. 8-2) с префиксом ASCII 254 (0FE hex или 11111110 двоичный). Модуль SLCD16x2 воспринимает байт после префикса как команду.

Пример: Для очистки дисплея, необходимо послать команду 00000001 в двоичном формате (или ASCII 1), посылайте [254] и [1] в модуль SLCD16x2 (скобки [ ] означают один байт).

COMMAND\DATA BIT	D7	D6	D5	D4	D3	D2	D1	D0
1. Initial LCD	0	0	0	0	0	0	0	0
2. Clear LCD	0	0	0	0	0	0	0	1
3. Return Home	0	0	0	0	0	0	1	*
4. Entry Mode Setting	0	0	0	0	0	1	I / D	S
5. Display Setting	0	0	0	0	1	D	C	B
6. Shift Display	0	0	0	1	S / C	R / L	*	*
7. Function Setting	0	0	1	*	N	F	*	*
8. Set CGRAM Address	0	1	A5	A4	A3	A2	A1	A0
9. Set DDRAM Address	1	A6	A5	A4	A3	A2	A1	A0

Standard instrction command set summary
(except Initial LCD is addition command.
Initialize make I/D=1, S=0, D=1, C=0, B=0, N=1, F=0, DDRAM Address=00

* Don't care bit

S 0=Automatic cursor shift after byte
1=Cursor not moved
I/D 0=After byte, decrease cursor position
1=After byte, increase cursor position
(when S=1, cursor won't be shifted .)

D 0=Display OFF, 1=Display ON
C 0=Cursor OFF, 1=Cursor ON
B 0=Cursor not blink, 1=Cursor blink

S/C 0=Cursor shift, 1=Display Shift
R/L 0=Left shift, 1=Right shift

N 0=1/8 Duty, 1=1/16 Duty
(not recommend to set this bit,
use jumper setting instead)
F 0=5x7 dot size, 1=5x10 dot size

A0 to A7 are CGRAM or DDRAM Address

Serial input timin g dia gram

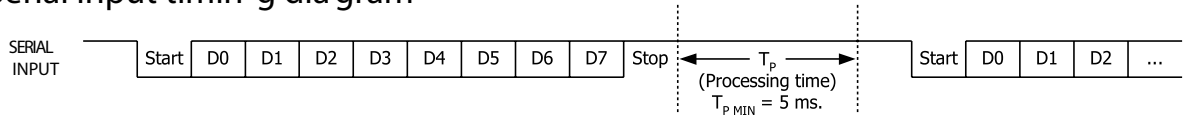
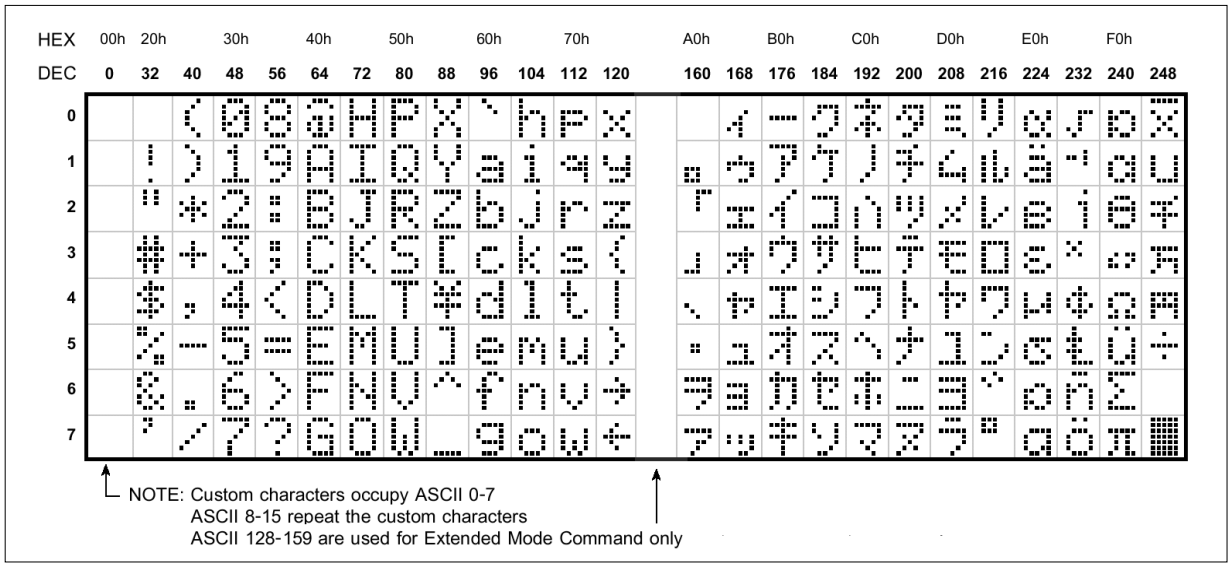


Рисунок 8-2 Команды модуля SLCD16x2

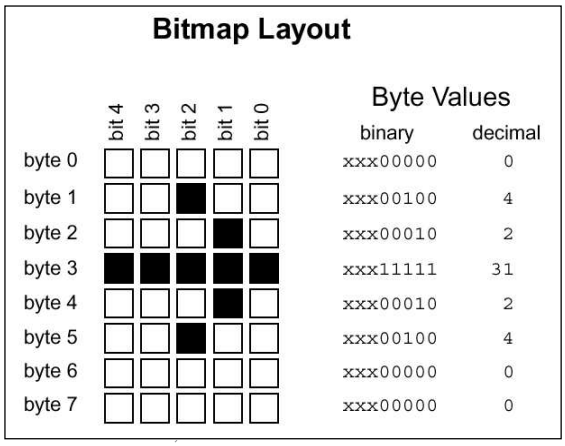
8.1.5 Набор символов

Большинство символов (на рисунке) не могут изменяться, потому что они хранятся в ПЗУ. Тем не менее, первые восемь символов, соответствующие ASCII 0...7, хранятся в ОЗУ. Эти символы могут быть перезаписаны пользователем.



Набор символов (для контроллера HD44780A или SED1278F0A)

Символы пользователя загружаются поочередно строками как показано на рисунке ниже. Один символ загружается за 8 циклов. Символ 0 расположен по адресам 00h-07h, символ 1 по адресам 08h-0Fh, символ 2 по адресам 10h-17h, ... символ 7 по адресам 38h-3Fh.

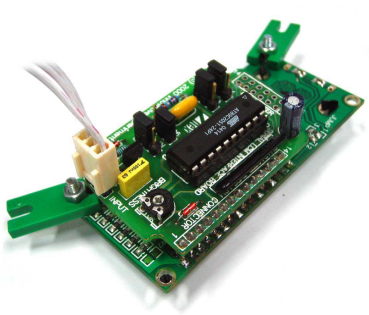


Example: Load arrow ymbol on CGRAM 3, aprogram would send the following bytes to the SLCD controller.

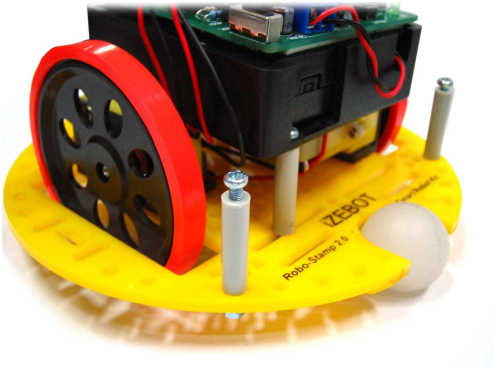
[254] , [01011000 b] , [0] , [254] , [01011001 b] , [4] , [254] , [01011010 b] , [2] , [254] , [01011011 b] , [31] , [254] , [01011100 b] , [2] , [254] , [01011101 b] , [4] , [254] , [01011110 b] , [0] , [254] , [01011111 b] , [0]

Задание 3
Подключение модуля SLCD16x2

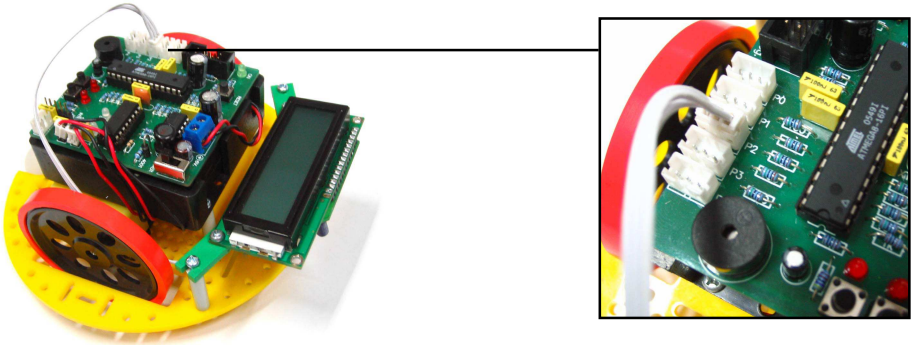
A3.1 Подключите сторону B-end кабеля JST3AB-8 к модулю SLCD16x2. Присоедините прямые крепления к отверстиям модуля SLCD16x2 винтами 3x10мм и гайками 3 мм .



A3.2 Вставьте винт 3x35мм в стойку 25мм и подключите к основанию MicroCamp гайками 3 мм как показано на рисунке .



A3.3 Присоедините модуль SLCD16x2 к стойкам. Затяните винты. Подключите конец A-end кабеля JST3AB к порту P2 на плате MicroCamp.



8.2 Библиотека soft_serout

Для облегчения вывода данных на дисплей SLCD16x25 можно использовать библиотеку, которая называется soft_serout.h.
Эта библиотека позволяет передавать данные любому устройству с последовательным интерфейсом в формате 8N1. Исходный код библиотеки показан на листинге 8-1.
Можно выбирать порт для связи (P0...P4). Библиотека содержит 3 функции.

8.2.1 Функция soft_serout_init()

Эта функция используется для настройки скорости передачи. Максимальная скорость 9600 бит в секунду.

Формат функции

```
void soft_serout_init(unsigned long baud_)
```

Пример:

```
soft_serout_init(9600);
```

8.2.2 Функция serout_byte()

Эта функция используется для выдачи байта через порт.

Формат функции

```
void serout_byte(char tx,unsigned char dat)
```

Здесь: tx порт MicroCamp. Может принимать значения 0...4 для портов P0...P4 соответственно

dat - пересылаемый байт данных

Пример:

```
serout_byte(2,0x80);
```

8.2.3 Функция serout_text()

Эта функция используется для выдачи нескольких байт через порт.

Формат функции

```
void serout_text(char tx,unsigned char * p)
```

Здесь: tx - номер порта. Значения от 0 до 4 для портов P0...P4

* p - посылаемая строка

Пример:

```
serout_text(2,"MicroCamp");
```

Передача сообщения "MicroCamp" через порт P2.

```
#include <avr/io.h>
#include <in_out.h>
#ifndef _soft_serout_
#define _soft_serout_

#define PRESCALER_1 (1<<CS20)               // (1/16M) 0.0625 us per MC
#define PRESCALER_8 (1<<CS21)               // (8/16M) 0.5 us per MC
#define PRESCALER_32 (1<<CS21) | (1<<CS20)  // (32/16M) 2 us per MC
#define PRESCALER_64 (1<<CS22)              // (64/16M) 4 us per MC
#define OFFSET_DELAY1  20                   // for out function used 20 us
#define OFFSET_DELAY2  18                   // for out function used 20 us
unsigned int base=0;
unsigned char base_start_rcv=0;
unsigned char TCCR2_cal=0;
unsigned int base;
unsigned int baud=9600;

void soft_serout_init(unsigned long baud_)     // Config.and Start up timer 2
{
   unsigned long tick=0;
   if(baud_<=4800)
   {
     tick = ((1000000/baud_)-OFFSET_DELAY1)/4;  // Calculate Delay for baudrate
     TCCR2_cal = PRESCALER_64;
   }
   else if(baud_>4800 && baud_<=9600)
   {
     tick = ((1000000/baud_)-OFFSET_DELAY2)/2;  // Calculate Delay for baudrate
     TCCR2_cal = PRESCALER_32;
   }

   TCCR2 = 0;                                   // Stop timer
   TIFR |= 1<<TOV2;                             // Ensure Clear Overflow flag
   base = 255-tick;
   base_start_rcv = 255-(tick/2);
}

// Delay For baudrate

void delay_baud(unsigned int _tick)
                {
   TCNT2 = _tick;                   // Load Prescaler form calculate
   TCCR2 = TCCR2_cal;               // Load interval
   while(!(TIFR & (1<<TOV2)));      // Wait until count success
   TIFR |= 1<<TOV2;                 // Ensure Clear Overflow flag
   TCCR2 = 0;                       // Stop timer 2
}

// Send Data  1 Byte
void serout_byte(char tx,unsigned char dat)
{
   int i;
   out_c(tx,0);                     // start bit
   delay_baud(base);                // Delay for start bit
   for(i=0;i<8;i++)
   {
     out_c(tx,dat & 0x01);          // Send data bit
```

Листинг 8-1 Исходный код библиотеки soft_serout library (с продолжением)

```
     delay_baud(base);              // Delay for calculate base
     dat=dat>>1;                    // Shift for next bit
   }
   out_c(tx,1);                     // stop bit
   delay_baud(base);                // Delay for stop bit
}

// Send More Than 1 byte

   void serout_text(char tx,unsigned char *
                                       p) {
                                  while(*p)
                                          {
                       serout_byte(tx,*p++);
                                          }
                                          }

#endif
```

Листинг 8-1 Исходный код библиотеки soft_serout library (окончание)

Задание 4
Простое программирование SLCD16x2

A4.1 Создайте новый проект и введите программу C в соответствии с листингом A4-1.

A4.2 Добавьте библиотеку soft_serout library в проект. Скомпилируйте проект.

A4.3 Подключите программатор PX-400 к плате MicroCamp-робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A4.4 Отключите питание и отсоедините ISP-кабель.

A4.5 Включите питание и наблюдайте работу SLCD16x2.

Модуль SLCD16x2 отобразит MicroCamp в первой строке.


```
#include <soft_serout.h>
#include <sleep.h>

void main()                          // Главная программа
{
     sleep(1000);                    // Пауза 1 секунда
     soft_serout_init(9600);         // Конфигурирование 9600 8N1
     serout_text(2,"MicroCamp");     // Посылка текста "MicroCamp" в SLCD
     while(1);                       // Стоп
}
```

Описание программы

В этом коде используются 2 библиотеки: soft_serout. h для передачи данных по последовательному каналу и sleep.h для пауз. Программа выполняет следующие действия:

1. Пауза 1 сек. для инициализации SLCD16x2.
2. Установка скорости 9600 бит в секунду.
3. Посылка сообщения “MicroCamp” в порт P2 для отображения на SLCD16x2.
4. Цикл.

Листинг A4-1 Программа для вывода теста на дисплей SLCD16x2

Задание 5

Управление дисплеем SLCD16x2 командами

Можно управлять дисплеем при помощи передачи на него специальных команд. В стандартном режиме необходимо передавать байт h 0xFE, а затем необходимую команду. Перечень команд приведен в начале главы.

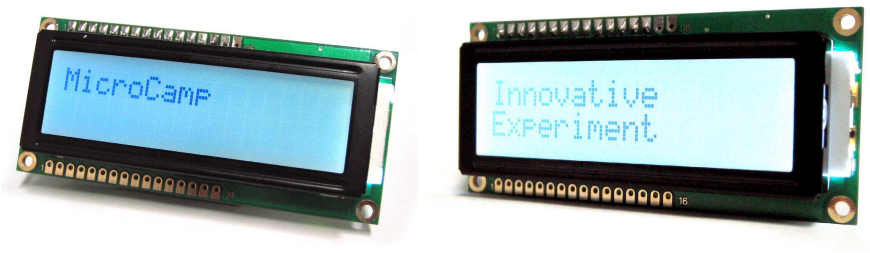
A5.1 Создайте новый проект и введите программу C в соответствии с листингом A5-1. Скомпилируйте проект.

A5.2 Подключите программатор PX-400 к плате MicroCamp-робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A5.3 Отключите питание и отсоедините ISP-кабель.

A5.4 Включите питание и наблюдайте работу SLCD16x2.

На дисплее SLCD16x2 будут показаны несколько сообщений.



```
#include <soft_serout.h>
#include <sleep.h>

void main()                                          // Главная программа
{
   unsigned char i=0;
   while (1)
   {
      sleep(1000);                                   // Пауза 1 секунда
1     soft_serout_init(2,9600);                      // Настройка порта 9600 8N1
      serout_byte(2,0xFE);serout_byte(2,0x00);       // Инициализация дисплея
      serout_byte(2,0xFE);serout_byte(2,0x80);       // Установка вывода в первый символ
      serout_text(2,"MicroCamp");                    // Вывод текста "MicroCamp"
2     serout_byte(2,0xFE);serout_byte(2,0xC0);       // Установка вывода во вторую строку
      serout_text(2,"Microcontroller");              // Вывод текста "Microcontroller"
      sleep(2000);
      serout_byte(2,0xFE);serout_byte(2,0x01);       // Очистка дисплея
      serout_byte(2,0xFE);serout_byte(2,0x85);       // Установка вывода в 5-й символ 1-й строки
3     serout_text(2,"From");                         // Вывод текста "From"
      sleep(500);
      serout_byte(2,0xFE);serout_byte(2,0x07);       // Сдвиг текста влево и уменьшение адреса
      for (i=0;i<9;i++)                              // 9 циклов
      {
4        serout_byte(2,0x20);                        // <<<<<< сдвиг данных влево
         sleep(200);
      }
      serout_byte(2,0xFE);serout_byte(2,0x05);       // Сдвиг текста вправо и увеличение адреса
      for (i=0;i<9;i++)
      {
5        serout_byte(2,0x20);                        // Сдвиг данных вправо 9 раз
         sleep(200);
      }
      for (i=0;i<9;i++)                              // Мигание 9 раз
      {
      serout_byte(2,0xFE);serout_byte(2,0x08);  // Отключение дисплея
6     sleep(200); serout_byte(2,0xFE);
      serout_byte(2,0x0C);                       // Включение дисплея
         sleep(200);
      }
      serout_byte(2,0xFE);serout_byte(2,0x00);       // Инициализация дисплея
      serout_byte(2,0xFE);serout_byte(2,0x80);       // Установка вывода в первый символ
7     serout_text(2,"Innovative");                   // Вывод текста "Innovative"
      serout_byte(2,0xFE);serout_byte(2,0xC0);       // Установка вывода в 1-й символ 2-й строки
      serout_text(2,"Experiment");                   // Вывод текста "Experiment"
   }
}
```

Листинг А5-1 Проверка дисплея SLCD16x2 в разных режимах

. Не вводите в программу разделительные линии (продолжение далее)

Описание программы

Часть 1 Инициализация модуля SLCD16x2.

Часть 2 Установка адреса вывода. В верхнюю строку (0x80) выводится сообщение MicroCamp. В нижнюю линию (0xC0) выводится Microcontroller.

Часть 3 Посылка команды очистки дисплея (0x01) и задание адреса вывода в 5-й символ верхней строки дисплея (0x85) для вывода From.

Часть 4 Посылка команды сдвиг влево (0x07) и зацикливание для сдвига From влево.

Часть 5 Посылка команды сдвиг вправо (0x05) и зацикливание для сдвига From обратно.

Часть 6 Посылка команд включения (0x08) и выключения (0x0C) дисплея в цикле. Это приведет к миганию сообщения From.

Часть 7 Действия аналогично Часть 2, но выводятся сообщения Innovative и в нижнюю строку Experiment.

Листинг A5-1 Проверка дисплея SLCD16x2 в разных режимах.

Не вводите в программу разделительные линии (окончание)

Глава 9

Движение робота MicroCamp по линии

В главе 7 было показано как считывать цифровые данные и контролировать движение робота. В этом задании будет необходимо считывать аналоговые данные и обрабатыватр их для различения черных и белых областей. Это необходимо для движения робота вдоль линии.

Робот MicroCamp оснащен 5 аналоговыми входами, подключенными к портам PC0...PC4 микроконтроллера ATmega8. Этот микроконтроллер оборудован АЦП 10-бит. На выходе АЦП данные принимают значения от 0 до 1,023 в десятичном формате.

Для упрощения программирования необходимо использовать библиотеку analog.h. Функции этой библиотеки позволяют настроить порт на считывание аналоговых данных, получить данные с АЦП и сохранить их в памяти. Результат может принимать значения от 0 до 1,023 в десятичном формате или от 0000H до 03FFH в шестнадцатиричном.

Для выполнения этого задания необходимо познакомиться с принципом действия ИК-детекторов. Они устанавливаются снизу круглой платформы робота. Эти детекторы используются для определения цвета подстилающей поверхности (черный и белый). Программирование робота для слежения за линией является классической задачей робототехники.

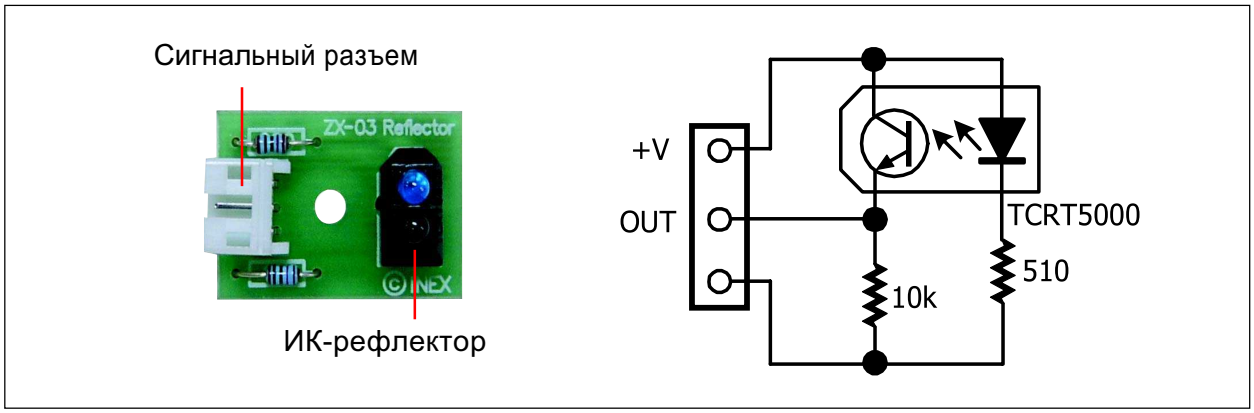


Рисунок 9-1 : ИК-рефлектор ZX-03

9.1 ИК-рефлектор ZX-03

Основой этого детектора является рефлективный сенсор TCRT5000. Он сконструирован для обнаружения ИК-отражений в ближней зоне. Излучателем является ИК-светодиод, приемником является ИК-транзистор. Когда излученный диодом ИК свет отражается от поверхности, он попадает на фототранзистор,что приводит к появлению тока через него. Чем больше света попадает на базу транзистора, тем больший ток протекает через него. Если этот детектор использовать как аналоговый датчик, то ZX-03 может различать оттенки серого цвета на бумаге или расстояние в небольшом диапазоне, если освещение остается постоянным.

Предпочтительное расстояние от 3 до 8 мм. Выходное напряжение меняется от 0.1 до 4.8В, что соответствует числовым данным от 20 до 1,000.

9.2 Функция analog: чтение аналогового сигнала в библиотеке analog.h

Эта функция используется для считывания аналоговых данных с портов PC0...PC5. Эти порты настраиваются как аналоговые входы.

Формат функции

```
unsigned int analog(unsigned char channel)
```

Параметр

channel：Выбор желаемого аналогового входа. Значение от 0 до 4,что соответствует портам P0...P4.

Возвращается значение от 0 до 1023.

Задание 6
Обнаружение черных и белых областей

Робот с подключенными ИК-детекторами готов к программированию.

Перед программированием движения по линии, необходимо запрограммировать робота на различение черных и белых областей.

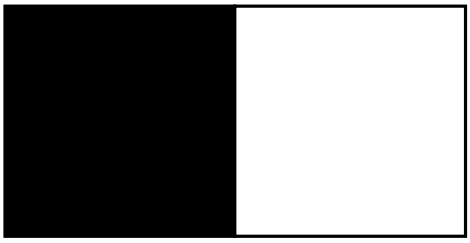
A6.1 Создайте новый проект и введите программу C в соответствии с листингом A6-1.

A6.2 Подключите библиотеку analog в проект. Скомпилируйте проект.

A6.3 Подключите программатор PX-400 к плате MicroCamp робота. Включите робота. Загрузите HEX-файл в память микроконтроллера.

A6.4 Отключите питание и отсоедините ISP-кабель.

A6.5 Сделайте тестовый черно-белый рисунок. Рисунок из двух квадратов - белый 30 x 30 см и черный 30 x 30 см как показано ниже.



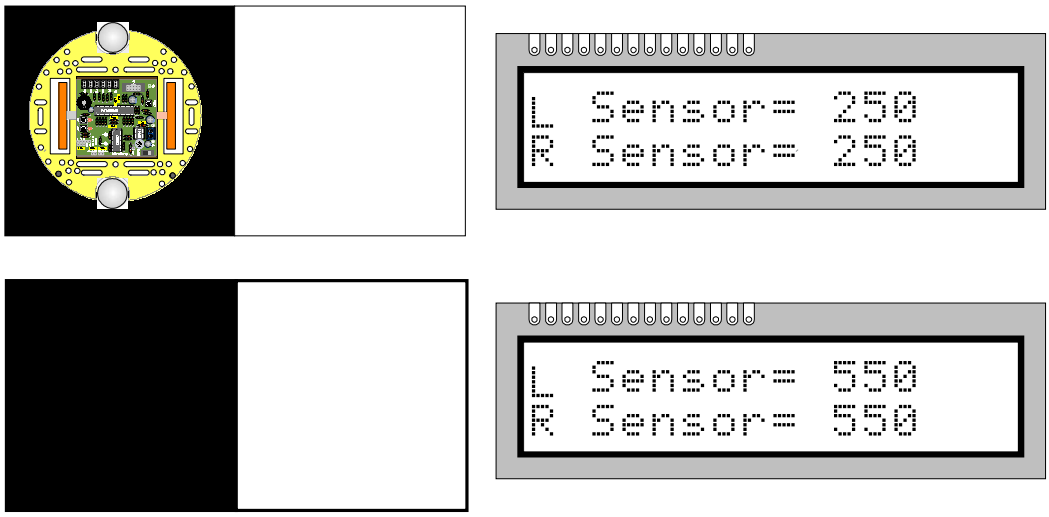
```
#include <stdlib.h>           // преобразование типов данных
#include <soft_serout.h>      // работа с жидкокристаллическим дисплеем
#include <analog.h>
#include <motor.h>
#include <sleep.h>
void lcd_command(unsigned char pin,unsigned char command)
{
  serout_byte(pin,0xFE);serout_byte(pin,command);
}
int main()
{
  unsigned int l_sensor=0,r_sensor=0;
  unsigned char dec1[4],dec2[4];
  sleep(1000);                        // Пауза 1 секунда
  soft_serout_init(2,9600);           // параметры порта 9600 8N1
  while(1)
  {
    l_sensor = analog(3);             // Считывание данных с левого сенсора
    r_sensor = analog(4);             // Считывание данных с правого сенсора
    utoa(l_sensor,dec1,10);           // преобразование Integer в десятичный Ascii
    utoa(r_sensor,dec2,10);           // преобразование Integer в десятичный Ascii
    lcd_command(2,0x80);              // установка адреса 1-й строки
    serout_text(2,"L Sensor=       ");
    lcd_command(2,0x8A);              // установка адреса 1-й строки
    serout_text(2,dec1);              // вывод Ascii данных
    lcd_command(2,0xC0);              // установка адреса 1-й строки
    serout_text(2,"R Sensor=       ");
    lcd_command(2,0xCA);              // установка адреса 1-й строки
    serout_text(2,dec2);              // вывод Ascii данных
    sleep(300);
  }
}
```

Описание программы

Микроконтроллер ATmega8 робота считывает данные с АЦП, подключенного к двум ИК-детекторам.

Листинг А 6-1 Проверка

A6.6 Установите запрограммированного робота (пункт A6.4) на белую поверхность тестового изображения. Включите робота. Запишите значения, выводимые на дисплей. Переместите робота на черную поверхность и запишите снова измеренные значения.



Результат:

На белой поверхности результат будет от 600 до 950

На черной поверхности результат будет от 100 до 300

Экспериментальное опорное значение для обнаружения будет

(600+100) /2 = 350.

Задание 7

Движение робота вдоль черной линии

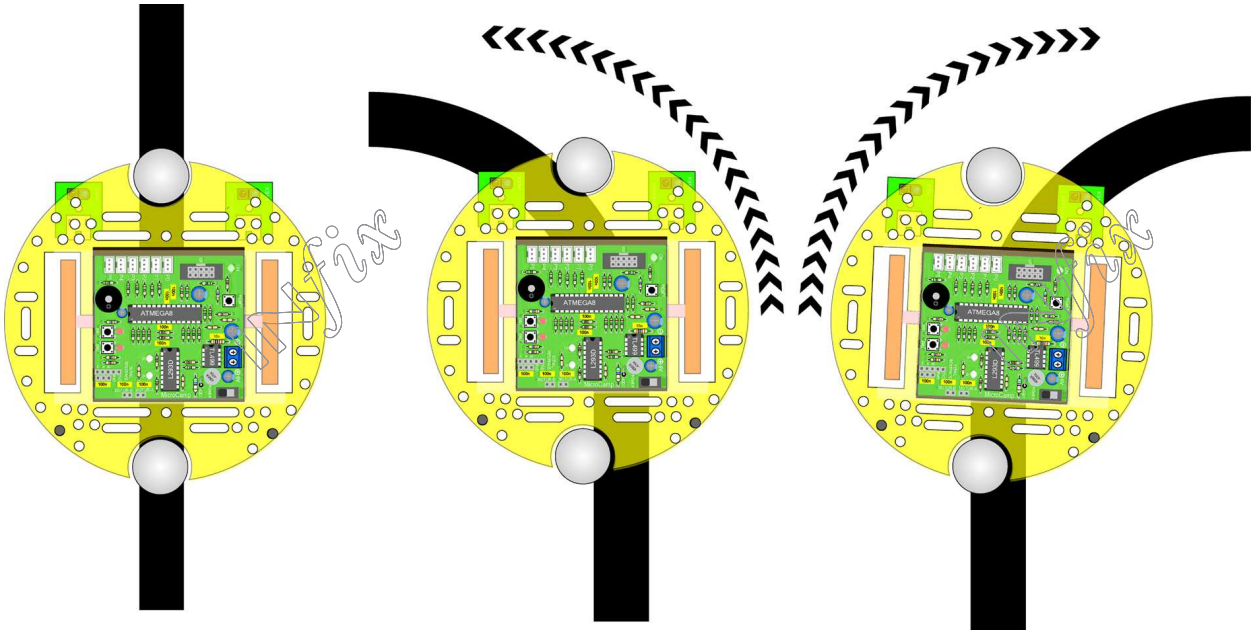
При движении робота по черной линии возможны три случая.

(1) Оба сенсора определяют белый цвет: робот будет двигаться вперед.

(2) Левый сенсор определяет черный цвет, а правый - белый. Это происходит, когда робот уклонился от линии вправо. Робот будет поворачивать влево для коррекции траектории.

(3) Левый сенсор определяет белый цвет, а правый - черный. Это происходит, когда робот уклонился от линии влево. Робот будет поворачивать вправо для коррекции траектории.

Введите программу, показанную на листинге A7-1



Случай-1 Случай-2 Случай-3

```
#include <in_out.h>
#include <sound.h>
#include <analog.h>
#include <motor.h>                              // управление моторами
unsigned int AD0=350,AD=350;                    // задание опорных значений

void main()
{
   while((in_d(2)==1));                         // ожидание нажатия SW1 для запуска

   while(1)
   {
   if((analog(3)>AD0)&&(analog(4)>AD1))    // оба сенсора - белый цвет

         forward(100);                          // движение вперед
     if (analog(0)<AD0)                         // левый сенсор - черный цвет
         s_left(100);                           // поворот влево
     if (analog(1)<AD1)                         // правый сенсор - черный цвет
         s_right(100);                          // поворот вправо
   }
}
```

Листинг A7-1 : Программа C для движения робота вдоль черной линии

A7.1 Сделайте тестовую поверхность с черной линией. Для этого можно использовать черную изоленту наклееную на белую поверхность. Вся поверхность за исключением линии пути должна быть белой. Также можно использовать черный маркер.

A7.2 Создайте новый проект и введите программу с листинга A7-1. Откомпилируйте проект.

A7.3 Подключите программатор PX-400 к плате MicroCamp-робота и загрузите HEX-файл в микроконтроллер. Выключите питание и отсоедините ISP-кабель.

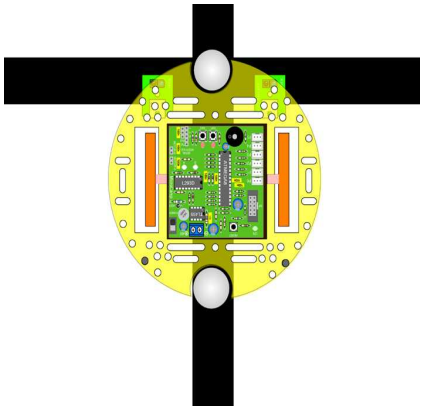
A7.4 Установите робота на черную линию. Включите питание и нажмите кнопку SW1.

Робот будет двигаться вдоль черной линии. Возможно робот потеряет линию. Вы можете увеличить точность слежения путем редактирования опорного значения сенсоров и механического изменения их ориентации.

Задание 8
Обнаружение пересечения линии

Вы можете редактировать программу из задания 7. Необходимо усовершенствовать робота MicroCamp таким образом, чтобы он при движении вдоль черной линии обнаруживал пересечение траектории при помощи тех же двух , сенсоров Необходимо только редактировать программу. Изменения конструкции робота не требуются.

Когда робот наезжает на пересекающую линию, оба детектора определяют черный цвет. Необходимо добавить обработку такого случая в программу. Корректированная программа показана на листинге A8-1.



A8.1 Измените линию траектории из задания 7. Добавьте несколько пересекающих линий. Их количество может быть произвольным. Расстояние между пересекающими линиями должно быть не менее двух длин робота.

A8.2 Создайте новый проект и введите программу с листинга A8-1. Откомпилируйте проект.

A8.3 Подключите программатор PX-400 к плате MicroCamp-робота и загрузите HEX-файл в микроконтроллер. Выключите питание и отсоедините ISP-кабель.

A8.4 Установите робота на черную линию. Включите питание и нажмите кнопку SW1.

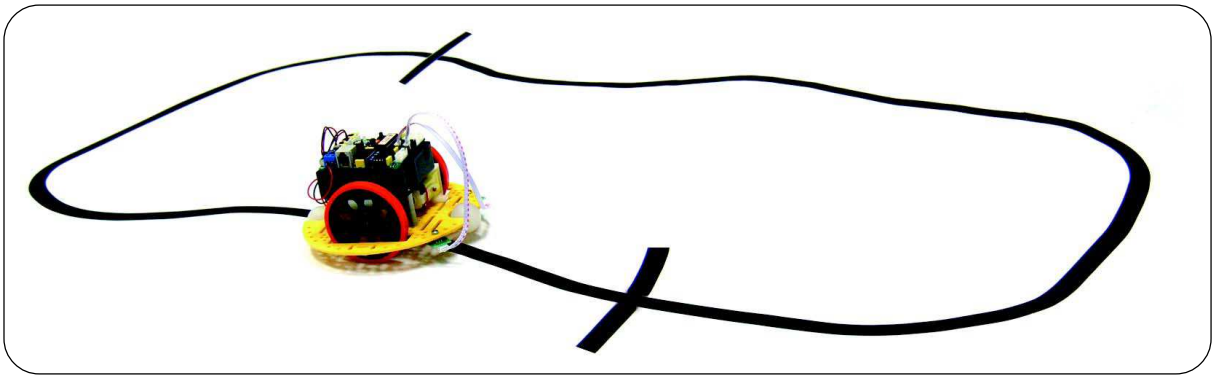
Робот будет двигаться вдоль черной линии. Когда робот обнаружит пересечение, он остановится и выдаст один звуковой сигнал. При обнаружении второго пересечения, робот выдаст два звуковых сигнала и число сигналов будет увеличиваться линейно.

Внимание: При остановке робота колеса блокируются мгновенно. Однако, этого недостаточно. Необходимо немного сдвигать робота назад. В этом случае он точно остановится на пересечении.

```
#include <in_out.h>
#include <sound.h>
#include <analog.h>
#include <motor.h>                              // контроль моторов
unsigned int AD0=350,AD1=350;                   // опорное значение датчиков
unsigned char i=0,j=0;                          // счетчик пересечений
void main()
{
   while((in_d(2)==1));                         // ожидание нажатия SW1 для запуска

   while(1)
   {
   if((analog(3)<AD0)&&(analog(4)<AD1))    // обнаружение пересечения
   {
   j++;
   backward(30);                                // сдвиг назад

        sleep(10);
        motor_stop(ALL);                        // торможение
        for (i=0;i<j;i++)                       // цикл обнаружения пересечения
        {
           sound(2500,100);
           sleep(50);
        }                                       // звуковой сигнал
        forward(100);                           // съезд с пересечения
        sleep(300);
     }
     if((analog(3)>AD0)&&(analog(4)>AD1))       // оба детектора - белый цвет
        forward(100);                           // движение вперед
     if (analog(3)<AD0)                         // левый детектор - черный цвет
        s_left(100);                            // поворот влево
     if (analog(4)<AD1)                         // правый детектор - черный цвет
        s_right(100);                           // поворот вправо
   }
}
```



Листинг А8-1 Программа движения вдоль черной линии и обнаружения пересечений.

Глава 10

Робот с ИК-измерителем дистанции

Одним из наиболее важных детекторов при построении робота является ИК-измеритель дистанции. В набор входит модуль GP2D120, который позволяет измерять дистанцию и обнаруживать препятствия при помощи инфракрасного света. Этот модуль позволяет создать робота, уклоняющегося от препятствий без физического контакта с ними.

10.1 Особенности модуля GP2D120*

- Используется отражение ИК-луча для измерения дистанции
- Может измерять дистанцию от 4 до 30 см
- Питание от 4.5 до 5 В, потребляемый ток 33 мА
- Выходное напряжение от 0.4 до 2.4 В при питании +5 В

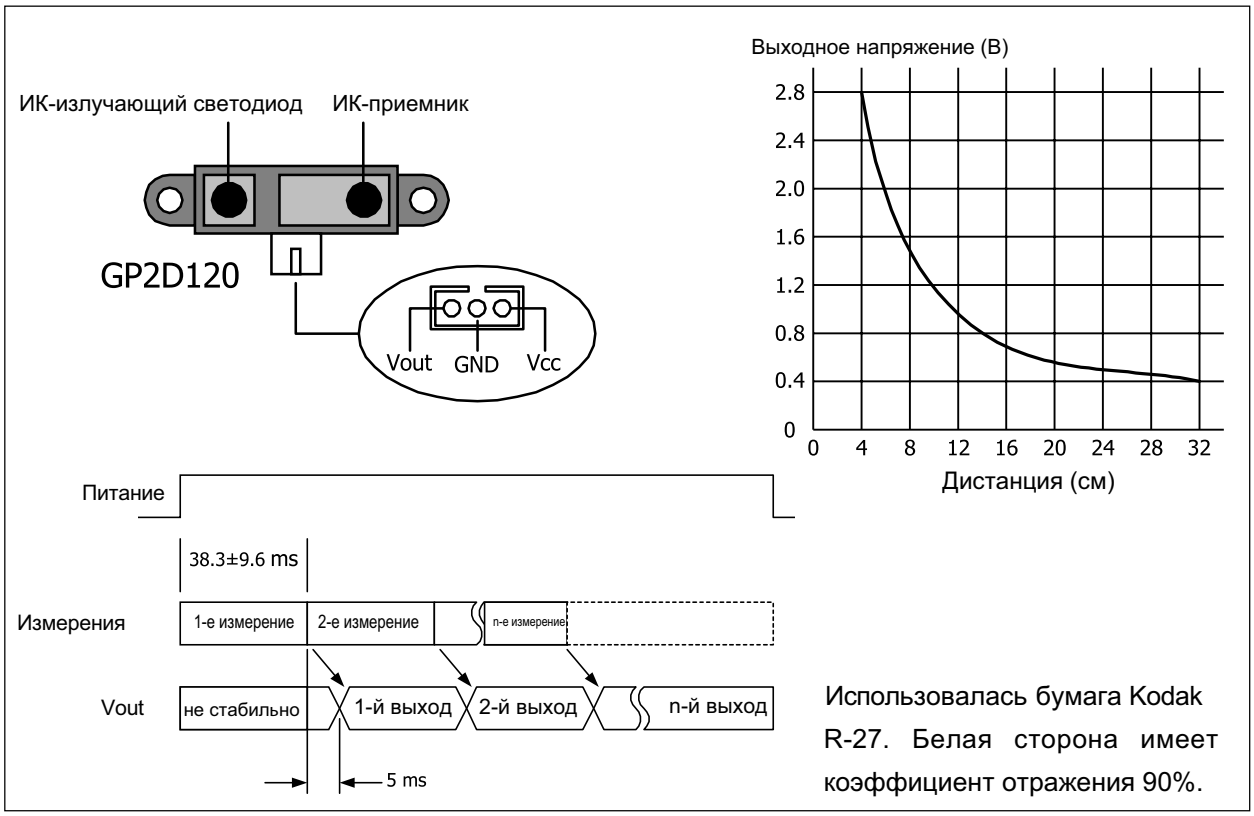


Рисунок 10-1 : Цоколевка GP2D120, характеристика датчика

* GP2D120 не входит в набор MicroCamp2.0 для начинающих, а поставляется в стандартном наборе.

Модуль GP2D120 имеет 3-контактный разъем: питание (Vcc), общий (GND) и выходное напряжение (Vout). Для получения корректных данных с модуля GP2D120 необходимо пропустить период инициализации порядка 32...52.9 мс.

Напряжение на выходе модуля GP2D120 для расстояния 30 см (питание +5 В) будет составлять от 0.25 до 0.55 В. Для дистанции 4 см выходное напряжение будет составлять 2.25 В ± 0.3 В.

10.2 Принцип действия ИК-измерителя дистанции

Измерение дистанции может осуществляться различными способами. Простейшим способом измерения дистанции является подсчет времени распространения ультразвука до препятствия и обратно. Это возможно сделать потому, что скорость распространения звука в воздухе относительно не высока и задержка может быть измерена имеющимся оборудованием. В случае инфракрасного света, время распространения луча до объекта и обратно очень мало и не может быть измерено существующим оборудованием. Однако, дистанцию можно измерить при помощи следующей теории.

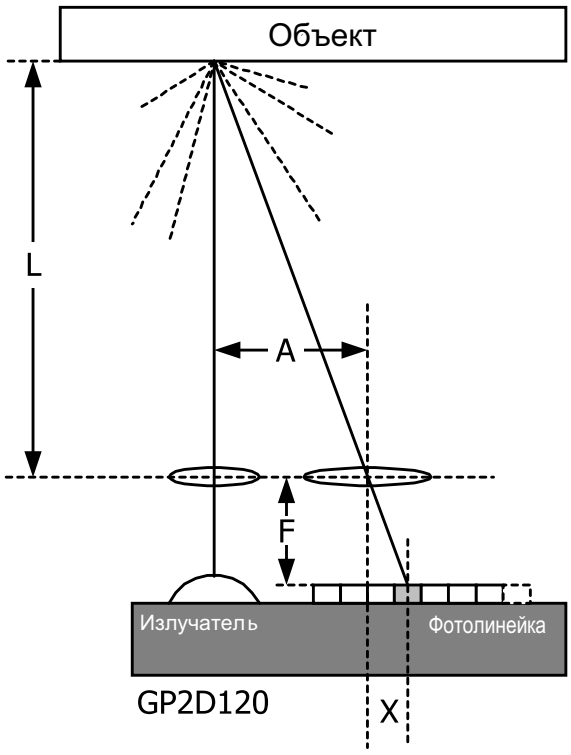
Инфракрасный свет излучается в направлении объекта через фокусирующую линзу, что позволяет сузить луч. Свет отражается от объекта, и часть отраженного света возвращается назад. Отраженный свет проходит через вторую линзу и попадает на линейку фото-транзисторов. Точка, в которую попадает отраженный луч, используется для вычисления дистанции до объекта (L) при помощи следующей формулы:

$$\frac{L}{A} = \frac{F}{X}$$

Таким образом,

$$L = \frac{F \times A}{X}$$

Измеренное значение дистанции преобразуется в постоянное напряжение, которое поступает на выход модуля.



10.3 Измерение напряжения на выходе GP2D120

Выходное напряжение модуля GP2D120 изменяется в зависимости от измеренной дистанции. Например, Vout = 0.5 В соответствует дистанции 26 см, а Vout = 2 V соответствует дистанции 6 см. В таблице 10-1 приведено соответствие выходного напряжения модуля GP2D120 измеряемой дистанции.

На выходе АЦП получаются числовые данные, которые будет необходимо конвертировать в дистанцию. Например, число 307 на выходе АЦП соответствует дистанции 8 см.

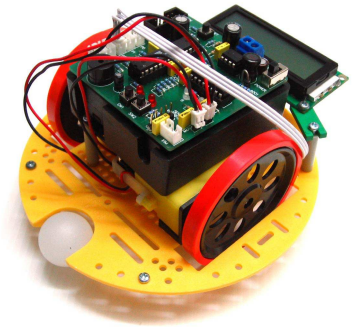
GP2D120™s Vout	Данные АЦП	Дистанция
0.4	82	33
0.5	102	26
0.6	123	22
0.7	143	19
0.8	164	16
0.9	184	14
1.0	205	13
1.1	225	12
1.2	246	11
1.3	266	10
1.4	287	9
1.5	307	8
1.6	328	8
1.7	348	7
1.8	369	7
1.9	389	6
2.0	410	6
2.1	430	6
2.2	451	5
2.3	471	5
2.4	492	5
2.5	512	5
2.6	532	4

Таблица 10-1: Соотношение между выходным напряжением модуля GP2D120™, данными на выходе АЦП и измеренной дистанцией .

Задание 9

Подключение модуля GP2D120 к роботу

Список деталей



Робот с подключенным дисплеем

Изогнутое крепление	x 2
Угловое крепление	x 2
Прямое крепление	x 4
Винт 3x10мм	x 4
Гайка 3мм	x 4

Модуль GP2D120 (поставляется в наборе MicroCamp2.0 Standard)

Внимание

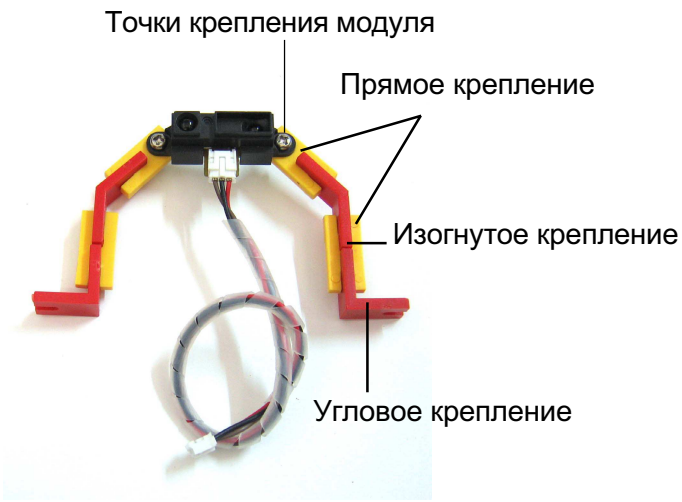
Модуль GP2D120 имеет цоколевку, отличающуюся от стандартной на плате MicroCamp. К модулю GP2D120 уже подключен специальный кабель, который не следует отключать. Необходимо только подключить свободный конец кабеля к плате MicroCamp. НЕ ОТКЛЮЧАЙТЕ кабель от модуля и не меняйте его на кабель от другого модуля.

A9.1 Закрутите 2 винта 3 x 10 мм в отверстия модуля GD2D120, используйте гайки 3мм. Не затягивайте винты.

A9.2 Подсуньте под гайки прямые крепления и затяните винты. (Не затягивайте слишком сильно, чтобы можно было двигать датчик)

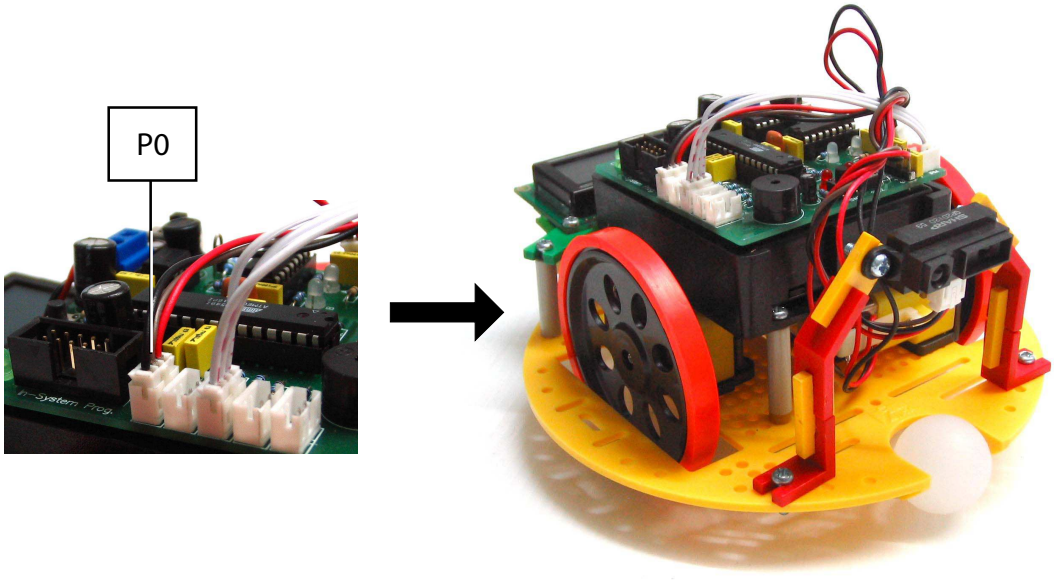
A9.3 Подключите изогнутое крепление к другому концу прямого.

A9.4 К изогнутому креплению подключите прямое. К прямому креплению подключите угловое.



A9.5 Отключите от робота контактные датчики. После этого присоедините модуль GP2D120 при помощи винтов 3 x 10 мм и гаек 3 мм.

A9.6 Подключите кабель модуля GPD120 к порту P0 робота как показано на рисунке ниже.



Задание 10
Измерение дистанции модулем GP2D120

Робот MicroCamp может считывать данные с модуля GP2D120 при помощи функции analog(). Для расчета дистанции необходимо использовать следующую формулу:

R = (2933 / (V + 20)) - 1

Здесь: R - дистанция в сантиметрах,

V - число на выходе АЦП. Диапазон от 0 до 1,023.

A10.1 Создайте новый проект и введите программу, приведенную на листинге A10-1.

A10.2 Подключите библиотеку analog к проекту. Скомпилируйте проект.

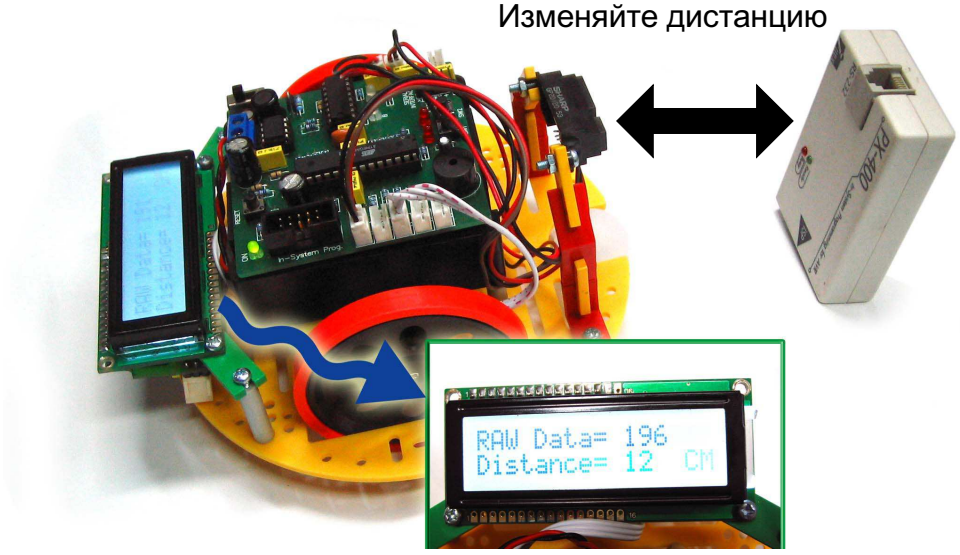
A10.3 Подключите программатор PX-400 к плате MicroCamp-робота и загрузите HEX-файл в микроконтроллер.

A10.4 Выключите питание и отсоедините ISP-кабель.

A10.5 Включите питание и разместите какой-нибудь объект перед модулем GP2D120. Наблюдайте за показаниями дисплея SLCD16x2.

A10.6 Изменяйте расстояние до объекта и контролируйте результат.

Убедитесь, что модуль GP2D120 может корректно измерять дистанцию от 4 до 30 см.



```
#include <stdlib.h>          // преобразование типов данных
#include <soft_serout.h>     // поддержка последовательного канала
#include <sleep.h>           // задержки
#include <analog.h>          // поддержка АЦП
#define m 2933               // константа для преобразования дистанции в см
#define b 20
#define k 1
void main()                  // главная программа
{
   unsigned char dec[4],dec2[4];    // для записи ascii после преобразования
   unsigned int gp2=0,cm=0;         // считывание данных с АЦП
   sleep(1000);                     // пауза 1 секунда
   soft_serout_init(2,9600);        // настройка последовательного порта 9600 8N1

   while(1)
   {
      gp2   = analog(0);            // чтение
      cm = (m/(gp2+b)) - k;         // преобразование данных в см
      utoa(gp2,dec,10);             // преобразование Integer в десят. Ascii
      utoa(cm,dec2,10);             // преобразование Integer в десят. Ascii
      serout_byte(2,0xFE);serout_byte(2,0x00);     // очистка дисплея
      serout_byte(2,0xFE);serout_byte(2,0x80);     // задание адреса 1-й строки
      serout_text(2,"RAW Data= ");
      serout_text(2,dec);                          // вывод данных Ascii
      serout_byte(2,0xFE);serout_byte(2,0xC0);     // задание адреса 2-й строки
      serout_text(2,"Distance=  ");
      serout_text(2,dec2);                         // вывод данных в см
      serout_byte(2,0xFE);serout_byte(2,0xCE);     // задание адреса 2-й
                                                   // строки 14-й позиции
      serout_text(2,"CM");
      sleep(500);
   }
}
```

Описание программы

(1) Настройка канала связи с дисплеем SLCD16x2.

(2) Считывание данных с порта P0 в цикле.

(3) Преобразование данных в см по формуле (m/(gp2+b)) - k.

(4) Преобразование в формат ASCII и вывод на дисплей SLCD16x2.

(5) Повтор измерений каждые 0,5 сек.

Листинг A10-1 : Программа для считывания данных с модуля GP2D120, преобразования их в см и вывод на дисплей SLCD16x2.

Задание 11

Бесконтактное уклонение от препятствий

A11.1 Создайте новый проект и введите программу, приведенную на листинге A11-1.

A11.2 Подключите библиотеку analog к проекту. Скомпилируйте проект.

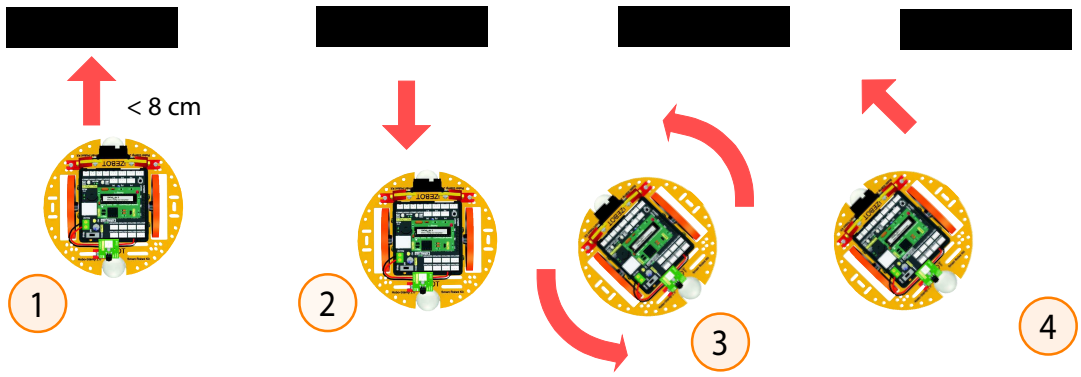
A11.3 Подключите программатор PX-400 к плате MicroCamp-робота и загрузите HEX-файл в микроконтроллер.

A11.4 Выключите питание и отсоедините ISP-кабель.

A11.5 Поставьте робота на пол. Включите питание робота и наблюдайте за его действиями.

A11.6 Обратите внимание на поведение робота при возникновении препятствия.

Робот проверяет наличие препятствий в диапазоне 8 см. Если препятствий нет, то робот двигается вперед. Если обнаруживается препятствие, то робот отъезжает назад, разворачивается влево и опять движется вперед.



```
#include <stdlib.h>                    // преобразование типов данных
#include <motor.h>                     // управление моторами
#include <sleep.h>                     // задержки
#include <sound.h>
#include <analog.h>                    // поддержка АЦП

void main()                            // главная программа
{
   unsigned int sensor=0;
   unsigned char i=0;
   sleep(200);sound(4000,50);          // запуск и звуковой сигнал
   while(1)
   {
      sensor=0;
      for (i=0;i<5;i++)
      {
         sensor=(sensor+analog(0));    // Считывание данных GP2D120 5 раз
      }
      sensor=(sensor/5);               // усреднение
      if (sensor>260)                  // до препятствия < 8 см ?
      {
         backward(50);sleep(800);      // объезд препятствия
         s_left(50);sleep(600);
      }
      else
      {
         forward(50);                  // движение вперед
      }
   }
}
```

Описание программы

(1) Начало работы со звуковым сигналом.

(2) Получение данных с модуля GP2D120 и запись в переменную. Усреднение по 5 измерениям.

(3) Проверка значение с АЦП больше 320 или нет. Если больше, это значит что до препятствия менее 8 см. Робот движется назад 0.8 секунды, затем разворачивается влево 0.6 секунды. Скорость 50%.

(4) Если измеренное значение меньше 320, то робот продолжает двигаться вперед.

(5) Повтор операций.

Листинг A11-1: Программа C для бесконтактного уклонения от препятствий при помощи ИК-модуля GP2D120.

Глава 11

Робот с дистанционным управлением

В главе 8 рассматривались вопросы связи через последовательный канал с модулем SLCD. Рассматривались, в основном, вопросы передачи данных. В данной главе будет рассмотрена техника приема данных. Будет использоваться новый компонент - пульт дистанционного управления ER-4. Пульт передает данные по последовательному каналу. Передаваемые данные модулируют несущую частоту 38 кГц. Робот должен быть оборудован ИК-приемником 38 кГц для демодулирования полученных данных.

Можно использовать пульт ER-4 для управления движением робота. Задания к этой главе позволят построить управляемого робота.

11.1 Пульт дистанционного управления ER-4*

- Рабочая дистанция от 4 до 8 метров на открытом пространстве.
- 4 кнопки для передачи команд.
- Малое энергопотребление.
- Питание 2.4-3.0 В от двух батареек AA.
- Передача последовательных данных в соответствии со стандартом RS-232 со скоростью 1200 bps и в формате 8N1.

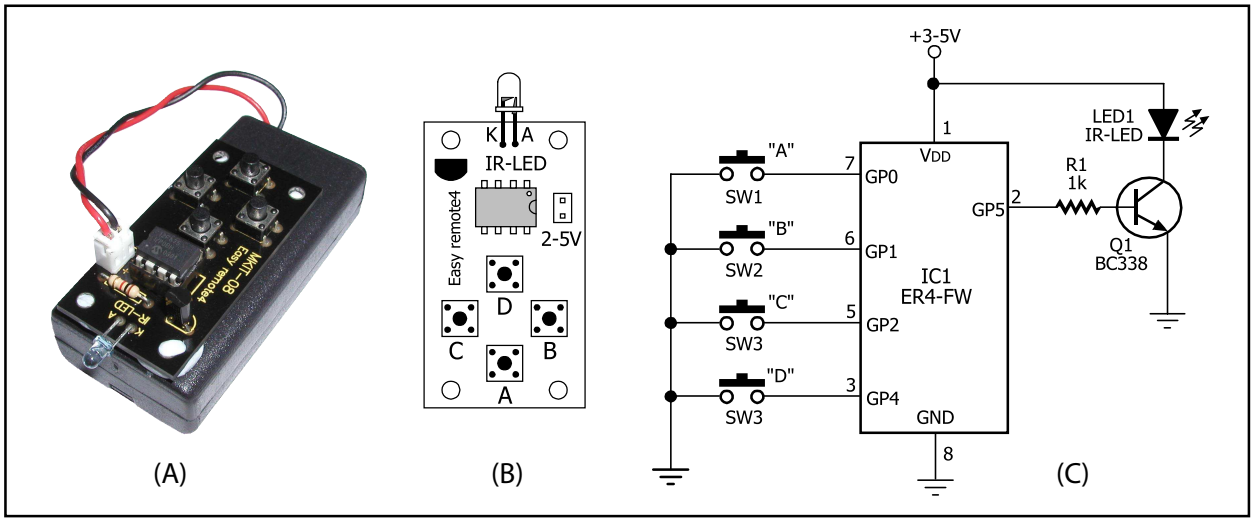


Рисунок 11-1 : Пульт дистанционного управления ER-4

* Пульт ER-4 и приемник ZX-IRM 38 кГц не включаются в набор MicroCamp2.0 Beginner. Поставляются в наборе Standard.

11.1.1 Формат передаваемых данных

Для упрощения работы с пультом ER-4, данные передаются по стандарту RS-232 со скоростью 1,200 bps и в формате 8N1. Данные передаются в соответствии с нажатой кнопкой на пульте. Месторасположение кнопок показано на рисунке 11-1.

При нажатии A передаются символы A, затем a.

При нажатии B передаются символы B, затем b.

При нажатии C передаются символы C, затем c.

При нажатии D передаются символы D, затем d.

Чередование букв сделано для различения серии быстрых нажатий и длинного нажатия кнопки. Если производятся повторяющиеся нажатия, то передается большая буква, при повторном нажатии передается маленькая. Если удерживать кнопку нажатой, то будет постоянно повторяться последняя буква.

11.2 ИК-приемник ZX-IRM

При передаче данных по ИК-каналу на большие расстояния (5...10 м) используется несущая частота 38 кГц. Таким образом, приемник должен демодулировать несущую 38 кГц. После этого передать последовательные данные в микроконтроллер.

Если несущая частота 38 кГц отсутствует, то на выходе будет логическая “1”. В противном случае, если детектируется 38 кГц, то на выходе будет логический “0”.

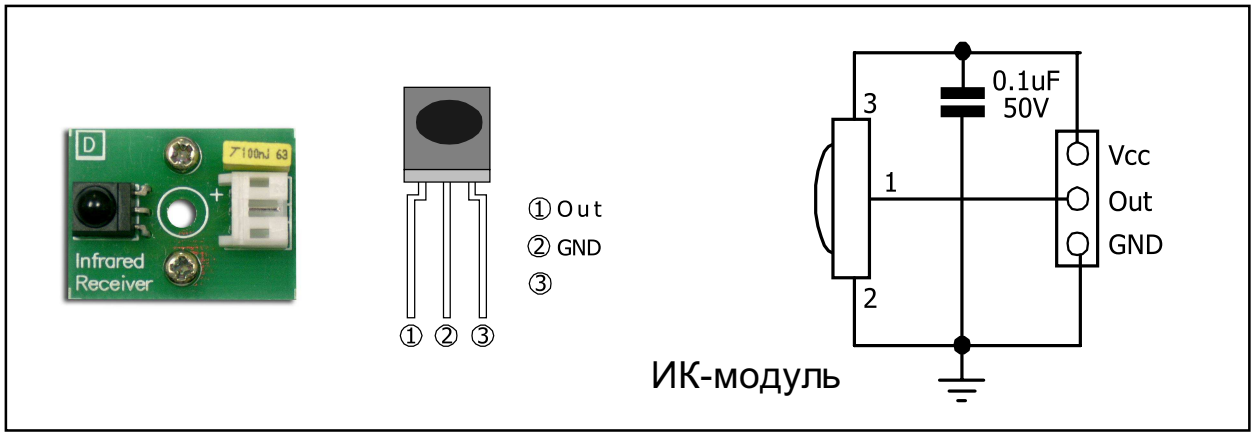


Рисунок 11-2: Модуль ИК-приемника 38 кГц

Задание 12

Подключение модуля ИК-приемника

Список деталей

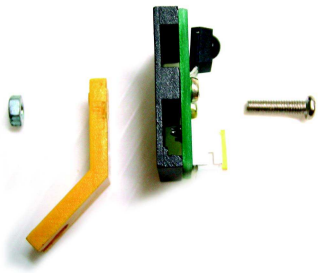
Винты 3x10 мм и
гайки 3 мм x 2

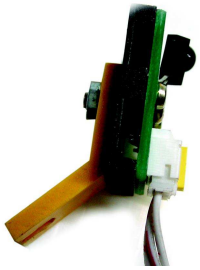
Угловое крепление x 1

Изогнутое крепление x 1

Модуль ИК-приемника
ZX-IRM 38 кГц x 1

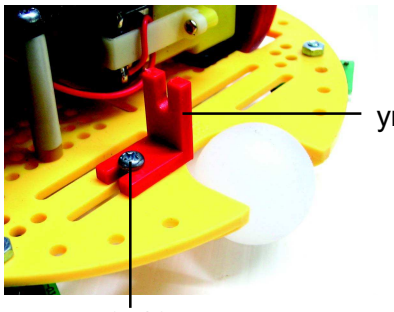
A12.1 Вставьте винт 3x10 мм через отверстие в модуле ZX-IRM и изогнутое крепление. Закрепите гайкой 3 мм как показано на рисунке.

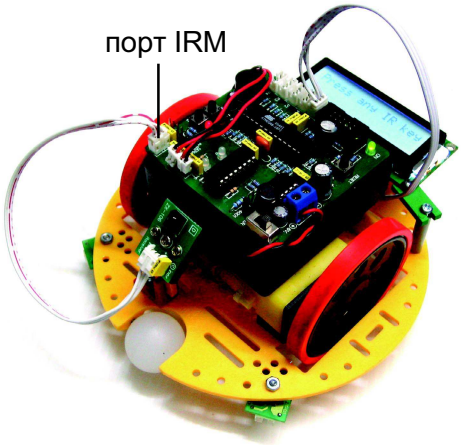




A12.2 Закрепите ИК-приемник с противоположной стороны от жидкокристаллического дисплея. Позиция может быть любой для обеспечения устойчивого приема ИК-сигналов от пульта ER-4.

A12.3 Подключите кабель приемника ZX-IRM в порт IRM-платы контроллера MicroCamp.





11.3 Прием данных по последовательному порту

Для упрощения программирования последовательного порта на языке C необходимо использовать специальную библиотеку serial.h file. Библиотека содержит функции, предназначеные для приема и передачи данных через канал UART микроконтроллера ATmega8.

В состав функций входят uart_set_baud() и uart_get_key(). Описание этих функций приведено ниже.

11.3.1 Функция uart_set_baud();

Задает скорость обмена через интерфейс UART микроконтроллера ATmega8.

Формат функции

```
void uart_set_baud(unsigned int baud)
```

Для работы с пультом ER-4 необходимо задавать скорость 1200 бит в секунду.

Пример: `uart_set_baud(1200;`

11.3.2 Функция uart_get_key();

Позволяет принять 1 байт данных через вход RxD микроконтроллера ATmega8. Можно задавать время до прекращения приема (timeout). Число 20,000 соответствует 30 мс.

Формат функции

```
char uart_getkey(unsigned int timeout)
```

Пример:

```
key=uart_getkey(20000);
```

Прием одного байта и запись в переменную key.

Задание 13

Прием данных, передаваемых пультом ER-4

A12.1 Создайте новый проект и введите программу, приведенную на листинге A12-1.

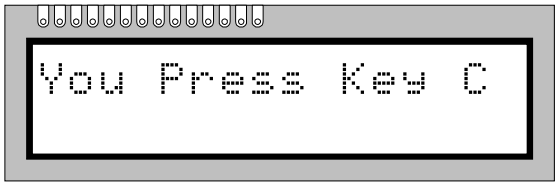
A12.2 Подключите библиотеку serial.h к проекту. Скомпилируйте проект.

A12.3 Подключите программатор PX-400 к плате MicroCamp робота и загрузите HEX-файл в микроконтроллер.

A12.4 Выключите питание и отсоедините ISP-кабель.

A12.5 Вставьте 2 батарейки AA в пульт дистанционного управления ER-4.

A12.6 Включите питание. Нажимайте кнопки на пульте ER-4 для передачи данных в приемник ZX-IRM робота MicroCamp. Наблюдайте за дисплеем SLCD.



```
#include <stdlib.h>                          // стандартная библиотека
#include <soft_serout.h>                     // модуль SLCD
#include <sleep.h>                           // задержка
#include <serial.h>                          // последовательный порт
#include <sound.h>                           // формирование звука
#include <motor.h>                           // контроль моторов

unsigned char key,flag=0;
unsigned char dec[4],bin[9];


    void main()
                                             // пауза 1 секунда
   soft_serout_init(0,9600);                 // настройка порта 9600 8N1
   uart_set_baud(1200);                      // скорость передачи данных пультом
   sound(2000,200);                          // звук
   serout_byte(0,0xFE);serout_byte(0,0x01);     // очистка дисплея
   serout_byte(0,0xFE);serout_byte(0,0x80);     // адрес первой строки
   serout_text(0,"You Press Key ");

   while(1)                                  // бесконечный цикл
   {
     key=uart_getkey(20000);                 // считывание данных с пульта timeout 33 мс
     if  ((key!=flag)&&(key>0x40)&&(key<0x7F))
                                             // проверка нажатой кнопки
     {
        serout_byte(0,0xFE);
        serout_byte(0,0x8E);                 // вывод на дисплей
        serout_byte(0,key);
        flag=key;
     }
   }
}
```

Описание программы

Для этой программы необходимо подключить 2 важные библиотеки serial.h и soft_serout.h. В программе выполняются следующие действия:

(1) Пауза 1 секунда для инициализации SLCD.

(2) Установка скорости работы с дисплеем SLCD 9,600 бит в секунду.

(3) Установка скорости работы с пультом ER-4 1,200 бит в секунду.

(4) Формирование стартового звука.

(5) Очистка дисплея и вывод сообщения "You Press Key".

(6) Ожидание нажатия кнопки на пульте ER-4. При отсутствии нажатия или приеме неизвестного кода программа игнорирует результат.

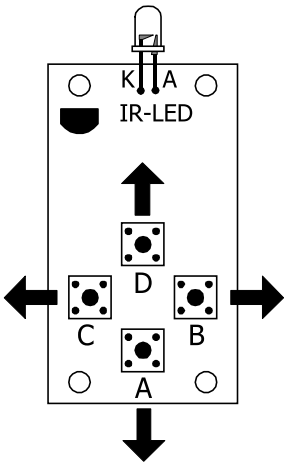
(7) Если код правильный, программа выводит код на дисплей.

Листинг A12-1 : Программа на языке C для приема данных с пульта ER-4 и вывода на дисплей .

Задание 14

Дистанционный контроль робота

В задании 12 робот MicroCamp принимает данные с пульта ER-4 8 значений таких, как A, B, C, D и a, b, c, d. Кнопки на пульте ER-4 сконструированы таким образом, чтобы направления движения робота соответствовали:



Вперед - верхняя кнопка - D или d

Назад - нижняя кнопка - A или a

Влево - левая кнопка - C или c

Вправо - правая кнопка - B или b

Все эти данные можно использовать для контроля движения робота.

A13.1 Создайте новый проект и введите программу, приведенную на листинге A13-1.

A13.2 Подключите библиотеку serial.h к проекту. Скомпилируйте проект.

A13.3 Подключите программатор PX-400 к плате MicroCamp-робота и загрузите HEX-файл в микроконтроллер.

A13.4 Выключите питание и отсоедините ISP-кабель.

```
#include <stdlib.h>                              // стандартная библиотека
#include <soft_serout.h>                         // модуль SLCD
#include <sleep.h>                               // задержка
#include <serial.h>                              // последовательный порт
#include <sound.h>                               // формирование звука
#include <motor.h>                               // контроль моторов
unsigned char key,flag=0;

void main()
{
   sleep(1000);                                  // пауза 1 секунда
   soft_serout_init(0,9600);                     // настройка порта 9600 8N1
   uart_set_baud(1200);                          // скорость передачи данных пультом

   sound(2000,200);                              // звук
   serout_byte(0,0xFE);serout_byte(0,0x01);      // очистка дисплея
   serout_byte(0,0xFE);serout_byte(0,0x80);      // адрес первой строки
   serout_text(0,"Press any IR Key");
   while(1)                                   // бесконечный цикл
   {
     key=uart_getkey(65000);                     // считывание данных с пульта timeout 33 мс
     if  ((key=='a')||(key=='A'))             // "A" движение назад
     {
        backward(100);
        if (flag!=1)                  // вывод "Backward" на дисплей
        {
           serout_byte(0,0xFE);
           serout_byte(0,0xC0);
           serout_text(0,"Backward   ");
           flag=1;
        }
     }
     else if ((key=='d')||(key=='D'))            // "D" движение вперед
     {
        forward(100);
        if (flag!=2)                             // вывод "Forward" на дисплей
        {
           serout_byte(0,0xFE);
           serout_byte(0,0xC0);
           serout_text(0,"Forward    ");
           flag=2;
        }
     }
     else if ((key=='c')||(key=='C'))            // "C" поворот влево
     {
        s_left(100);
        if (flag!=3)                             // вывод "Turn Left" на дисплей
        {
           serout_byte(0,0xFE);
           serout_byte(0,0xC0);
           serout_text(0,"Turn Left ");
           flag=3;
        }
     }
     else if ((key=='b')||(key=='B'))            // "B" поворот направо
```

Листинг A13-1: программа робота с дистанционным управлением

(продолжение на следующей странице)

```
      {
         s_right(100);
         if (flag!=4)                          //  вывод "Turn Right" на дисплей
         {
            serout_byte(0,0xFE);
            serout_byte(0,0xC0);
            serout_text(0,"Turn  Right");
            flag=4;
         }
      }
      else
      {motor_off();flag=0;}                    //  остановка моторов, если не нажата
   }                                                кнопка
}
```

Описание программы

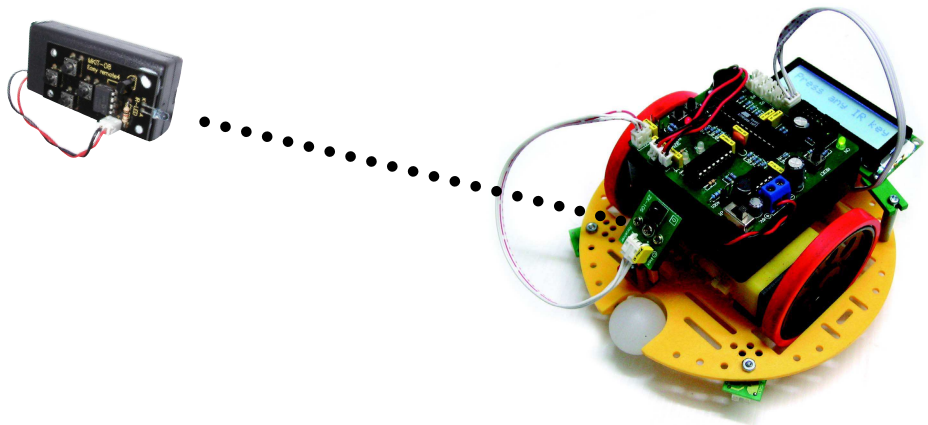
Эта программа разработана на основе предыдущей (листинг A12-1) для дистанционного управления движением робота. Программа контролирует нажатие кнопок на пульте и, соответственно, управляет моторами.

Также производится вывод сообщений на дисплей.

Листинг A13-1: программа робота с дистанционным управлением (окончание)

A13.6 Поставьте робота на пол. Включите питание.

A13.7 Нажимайте кнопки на пульте ER-4 для управления роботом. Направляйте пульт в сторону приемника. Наблюдайте за движением робота.



Исходные тексты программ библиотеки MicroCamp 2.0

in_out.h : Чтение и Запись цифровых данных по любому порту

```
#ifndef _IN_OUT_
#define _IN_OUT_
#define toggle_b(x) DDRB |= _BV(x); PORTB ^= _BV(x);
#define toggle_c(x) DDRC |= _BV(x); PORTC ^= _BV(x);
#define toggle_d(x) DDRD |= _BV(x); PORTD ^= _BV(x);
char in_b(char _bit)
{
DDRB &= ~(1<<_bit);
return((PINB & _BV(_bit))>>_bit);
}
char in_c(char _bit)
{
DDRC &= ~(1<<_bit);
return((PINC & _BV(_bit))>>_bit);
}
char in_d(char _bit)
{
DDRD &= ~(1<<_bit);
return((PIND & _BV(_bit))>>_bit);
}
void out_b(char _bit,char _dat)
{
DDRB |= _BV(_bit);
if(_dat)
PORTB |= _BV(_bit);
else
PORTB &= ~_BV(_bit);
}
void out_c(char _bit,char _dat)
{
DDRC |= _BV(_bit);
if(_dat)
PORTC |= _BV(_bit);
else
PORTC &= ~_BV(_bit);
}
void out_d(char _bit,char _dat)
{
DDRD |= _BV(_bit);
if(_dat)
PORTD |= _BV(_bit);
else
PORTD &= ~_BV(_bit);
}


#endif
```

sleep.h : Библиотека функции задержки

```
#ifndef _sleep_
#define _sleep_
void sleep(unsigned int ms)
{
unsigned int i,j;
for(i=0;i<ms;i++)
for(j=0;j<795;j++);
}
#endif
```

analog.h : Библиотека аналогового чтения входа

```
unsigned int analog(unsigned char channel)
{
unsigned int adc_val;
ADMUX = 0x40;
ADMUX |= channel; // Режим одиночного входа
ADCSRA = 0xC6;
while((ADCSRA & (1<<ADSC)));
adc_val = ADCL;
adc_val += (ADCH*256);
return(adc_val);
}
```

sound.h : Библиотека звукового генератора

```
#include <in_out.h>
#include <sleep.h>
void delay_sound(unsigned int ms)
{
unsigned int i,j;
for(i=0;i<ms;i++)
for(j=0;j<200;j++);
}
void sound(int freq,int time)
{
int dt=0,m=0; // Сохранить значение и
dt = 5000/freq; // Сохранить активную логическую задержку
time = (5*time)/dt; // Сохранить значение счетчика для генерации звука
for(m=0;m<time;m++) // Цикл генерации звука (Логический переключатель P0.12)
{
out_d(4,1);
delay_sound(dt); // Задержка для получения звука необходимой частоты
out_d(4,0);
delay_sound(dt); // Задержка для получения звука необходимой частоты
}
}
void sound_cnt(unsigned char cnt,int freq,int time)
{
unsigned char i;
for (i=0;i<cnt;i++)
{
sound(freq,time);
sleep(300);
}
```

led.h : Библиотека управления светодиодами

```
// Библиотека для управления светодиодным индикатором
// посредством прерывания от таймера Timer 2 каждые 5 мсек
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <in_out.h>
unsigned char LED=0;
unsigned char LED_cnt;
SIGNAL (SIG_OVERFLOW2) // Интервал 10 мсек
{
TCNT2 = 178; // Перезагрузка интервала 10 мсек(TCNT2 = 178)
LED_cnt++; // Увеличение значения Счетчика 10 мсек на единицу
if (LED_cnt>30) // Проверка значения Счетчика 10 мсек на величину 30
{
LED_cnt=0; // Очистка Счетчика
if (LED==1) // Проверка разрешена ли работа светодиода LED1
{
toggle_c(5);
}
else if (LED==2) // Проверка разрешена ли работа светодиода LED2
{
toggle_d(1);
}
else if (LED==3) // Проверка разрешена ли одновременная работа
                                          // светодиодов LED1 и LED2
{
toggle_c(5);
toggle_d(1);
}
}
}
void interval_init() // Конфигурирование и Запуск таймера Timer 0
{
TCCR2 |= (1<<CS22)|(1<<CS21)|(1<<CS20);
// Значение коэффициента деления предварительного делителя 1024,
// для тактовой частоты 16 МГц,
// 1 мсек = 1024/16МГц = 64мксек/значение счетчика
TIFR |= 1<<TOV2; // Установка бита TOV2 / Очистка
TIMSK |= 1<<TOIE2; // Разрешение Прерывания по Переполнению для таймера Timer2
TCNT2 = 178; // Интервал 10 ms
sei(); // Разрешить все прерывания
}
void led1_on() // Начало процесса мигания светодиода LED1
{
interval_init();
LED |= (1<<0) ;
}
void led1_off() // Окончание процесса мигания светодиода LED1
{
LED &= ~_BV(0);
}
void led2_on()
{
interval_init(); // Начало процесса мигания светодиода LED2
LED |= (1<<1) ;
```

```
}
void led2_off()
{
LED &= ~_BV(1); // Окончание процесса мигания светодиода LED2
}
```

motor.h : Библиотека управления двигателем постоянного тока

```
/* Конфигурация аппаратных средств
ДВИГАТЕЛЬ1
- PD7 Присоединено к порту 1B
- PD6 Присоединено к порту 1A
- PB1 Connect to 1E
ДВИГАТЕЛЬ 2
- PB0 Присоединено к порту 2A
- PD5 Присоединено к порту 2B
- PB2 Присоединено к порту 2E */
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#define ALL 3 // Очистить весь двигатель
#define all 3 // Очистить весь двигатель
unsigned char _duty1=0,_duty2=0; // Переменные, в которых запоминается
    // значение скважности импульсов
    // управления каждым из двигателей
char pwm_ini =0; // Флажок для начальной проверки
SIGNAL (SIG_OVERFLOW1) // Обработчик прерывания по переполнению. Интервал 1
мсек
{
OCR1AL = _duty1; // Чтение значения скважности 1
OCR1BL = _duty2; // Чтение значения скважности 2
}
void pwm_init()
{
TCCR1A |= (1<<WGM10);
TCCR1B = (1<<CS12)|(1<<CS10)|(1<<WGM12); // Настройка предварительного делителя
// TCCR1B = (1<<CS12)|(1<<WGM12); // Настройка предварительного делителя
TIFR |= 1<<TOV1; // Установка бита TOV0 / clear
TIMSK |= 1<<TOIE1; //Разрешение прерывания по переполнению таймера Timer0
//timer_enable_int(_BV(TOIE1));
sei();
}
void pwm(char channel,unsigned int duty)
{
duty = (duty*255)/100; // Преобразование значения в интервале 0-100
                       // в значение в интервале 0-255
if(pwm_ini==0) // Значение для ШИМ начальное ?
{
pwm_init(); // Если значение начальное, то проводим инициализацию
pwm_ini=1;  // и устанавливаем флажок инициализации
}
if(channel==2)
{
TCCR1A |= _BV(COM1A1);
DDRB |= _BV(PB1);
OCR1AL = duty;
_duty1 = duty;
```

```
}
else if(channel==1)
{
TCCR1A |= _BV(COM1B1);
DDRB |= _BV(PB2);
OCR1BL = duty;
_duty2 = duty;
}
else if(channel==3)
{
TCCR1A |= _BV(COM1A1);
DDRB |= _BV(PB1);
OCR1AL = duty;
_duty1 = duty;
TCCR1A |= _BV(COM1B1);
DDRB |= _BV(PB2);
OCR1BL = duty;
_duty2 = duty;
}
}
void motor(char _channel,int _power)
{
if(_power>0)
{
pwm(_channel,_power);
if(_channel==2)
{
out_d(7,1);
out_d(6,0);
}
else if(_channel==1)
{
out_d(5,0);
out_b(0,1);
}
}
else
{
pwm(_channel,abs(_power));
if(_channel==2)
{
out_d(7,0);
out_d(6,1);
}
else if(_channel==1)
{
out_d(5,1);
out_b(0,0);
}
}
}
void motor_stop(char _channel)
{
pwm(_channel,100);
if(_channel==2 ||_channel==3)
{
```

```
out_d(7,0);
out_d(6,0);
}
if(_channel==1||_channel==3)
{
out_d(5,0);
out_b(0,0);
}
}
void motor_off()
{
pwm(3,0);
out_d(7,0);
out_d(6,0);
out_d(5,0);
out_b(0,0);
}
void forward(int speed)
{
motor(1,speed);
motor(2,speed);
}
void backward(int speed)
{
motor(1,speed*-1);
motor(2,speed*-1);
}
void s_left(int speed)
{
motor(1,speed);
motor(2,speed*-1);
}
void s_right(int speed)
{
motor(1,speed*-1);
motor(2,speed);
}
```

timer.h : Timer library

```
#include <C:/WinAVR/avr/include/avr/interrupt.h>
#include <C:/WinAVR/avr/include/avr/signal.h>
/************ Обработчик прерывания от таймера Timer0 *****************/
/****************** Интервал 1 мсек ***********************************/
unsigned long _ms=0;
SIGNAL (SIG_OVERFLOW0) // Интервал 1 мсек
{
TCNT0 = 6; // Интервал 1 мсек
_ms++;
}
void timer_start(void) // Конфигурирование и запуск таймера Timer0
{
TCCR0 = (1<<CS01)|(1<<CS00); // Предварительный делитель 64,16МГц,
                             // 1 мсек = 64/16M = 4мксек/значение счетчика
TIFR |= 1<<TOV0; // Установка бита TOV0 / clear
TIMSK |= 1<<TOIE0; // Разрешение прерывания по переполнению таймера Timer0
TCNT0 = 6; // Интервал 1 мсек
```

```
sei(); // Разрешение всех прерываний
_ms = 0;
}
void timer_stop()
{
TCCR0 = 0; // Остановка таймера
TCNT0 = 0;
TIMSK &= ~_BV(TOIE0); // Очистка бита TOIE0
_ms = 0; // Очистка значения переменной времени
}
void timer_pause()
{
TCCR0 = 0; // Остановка таймера без очистки значения переменной времени
}
void timer_resume()
{
TCCR0 = (1<<CS01)|(1<<CS00); // Предварительный делитель 64,16МГц,
                             // 1 мсек = 64/16M = 4мксек/значение счетчика
}
unsigned long msec()
{
return(_ms);
}
unsigned long sec()
{
return(_ms/1000);
}
```

serial.h : Библиотека последовательного обмена данными через UART ATmega8

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define F_OSC 16000000 /* Частота тактового генератора в Гц */
#define UART_BAUD_CALC(x,F_OSC) ((F_OSC)/((x)*16l)-1)
#define even_uart_rec() SIGNAL(SIG_UART_RECV)
//---- Условная проверка типа данных параметров для отображения ----//
#ifndef TEST_CHAR_TYPE(x)
#define TEST_CHAR_TYPE(x) *x=='%' && (*(x+1)=='c' || *(x+1)=='C')
#endif
#ifndef TEST_INT_TYPE(x)
#define TEST_INT_TYPE(x) *x=='%' && (*(x+1)=='d' || *(x+1)=='D')
#endif
#ifndef TEST_LONG_TYPE(x)
#define TEST_LONG_TYPE(x) *x=='%' && (*(x+1)=='l' || *(x+1)=='L')
#endif
#ifndef TEST_FLOAT_TYPE(x)
#define TEST_FLOAT_TYPE(x) *x=='%' && (*(x+1)=='f' || *(x+1)=='F')
#endif
#ifndef TEST_STRING_TYPE(x)
#define TEST_STRING_TYPE(x) *x=='%' && (*(x+1)=='s' || *(x+1)=='S')
#endif
```

```
#ifndef F_PREC
#define F_PREC 3
#endif
unsigned int _baud=9600;
char uart_ini=0;
char _key=0;
#ifndef USE_EVEN_UART_REC
SIGNAL(SIG_UART_RECV)
{
_key = UDR;
}
#endif
void uart_set_baud(unsigned int baud)
{
_baud = baud;
uart_ini==1;
// Установка скорости обмена
UBRRH = (unsigned int)(UART_BAUD_CALC(baud,F_OSC)>>8);
UBRRL = (unsigned int)UART_BAUD_CALC(baud,F_OSC);
// UBRRH = 00; //для скорости 9600 бит/сек
// UBRRL = 51;

// Разрешение работы приемника и передатчика; разрешение прерывания RX
UCSRB |= (1 << RXEN) | (1 << TXEN) | (1 << RXCIE);
// Асинхронный режим 8 бит данных 1 стартовый бит (8N1)
UCSRC |= (1 << URSEL) | (3 << UCSZ0);
sei(); // Разрешение всех прерываний
}
unsigned int uart_gets_baud()
{
return(_baud);
}
void uart_putc(unsigned char c)
{
if(uart_ini==0)
{
uart_ini==1;
uart_set_baud(_baud);
}
while(!(UCSRA & (1 << UDRE)));
UDR = c; // передача символа
}
void uart_puts(char *s)
{
while (*s)
{
uart_putc(*s);
s++;
}
}
void uart(char *p,...)
{
char *arg,**pp; // Указатель на указатель
char *ptr,buff[16]; // s_arg_offset=0,s_arg_i=0//;
pp = &p;
```

```
ptr = p; // Копирование адреса
arg = pp; // Копирование адреса точки p
arg += 2; // Перемещение на 2 позиции для получения
          // точки расположения первого параметра
if(uart_ini==0)
{
uart_ini==1;
uart_set_baud(_baud);
}
while(*ptr) // Проверка: ссылка на данные = 0?
{
if(TEST_CHAR_TYPE(ptr))
{
uart_putc(toascii(*arg));
arg+=2; // Вычисление адреса для символьного типа данных
ptr++; // Вычисление адреса для параметра %d
}
else if(TEST_INT_TYPE(ptr))

{
p = ltoa(*(unsigned int *)arg,&buff[0],10);
uart_puts(p);
arg+=2; // Вычисление адреса для типа данных int
ptr++; // Вычисление адреса для параметра %i
}
else if(TEST_LONG_TYPE(ptr))
{
p = ltoa(*(long *)arg,&buff[0],10);
uart_puts(p);
arg+=4; // Вычисление адреса для типа данных long
ptr++; // Вычисление адреса для параметра %l
}
else if(TEST_FLOAT_TYPE(ptr))
{
p = dtostrf(*(float *)arg,2,F_PREC,&buff[0]);
// Преобразование вещественного значения в строку символов
// (с использованием библиотеки libm.a)
uart_puts(p);
arg+=4; // Вычисление адреса для типа данных long
ptr++; // Вычисление адреса для параметра %l
}
else
{
uart_putc(*ptr); // Отправка строки данных на ЖКИ
}
ptr++; // Однократное увеличение значения адреса
}
}
/*
char uart_getkey()
{
char _c=_key;
if(uart_ini==0)
{
uart_ini==1;
uart_set_baud(_baud);
```

```
}
_key = 0;
return(_c);
}
*/
char uart_getkey(unsigned int timeout)

{
unsigned int cnt=1;
char _c=0;
if(uart_ini==0)
{
uart_ini==1;
uart_set_baud(_baud);
}
while(!_key&&(cnt<timeout))
{cnt++;}
_c = _key;
_key = 0;
return(_c);
}
```

soft_serout.h : Библиотека последовательного вывода данных для любого порта микроконтроллера ATmega8 (без использования модуля UART)

```
#include <avr/io.h>
#include <in_out.h>
#ifndef _soft_serout_
#define _soft_serout_
#define PRESCALER_1 (1<<CS20) // (1/16M) 0.0625 мксек на мсек
#define PRESCALER_8 (1<<CS21) // (8/16M) 0.5 мксек на мсек
#define PRESCALER_32 (1<<CS21) | (1<<CS20) // (32/16M) 2 мксек на мсек
#define PRESCALER_64 (1<<CS22) // (64/16M) 4 мксек на мсек
#define OFFSET_DELAY120 // для функции вывода используется 20 мксек
#define OFFSET_DELAY218 // для функции вывода используется 18 мксек
unsigned int base=0;
unsigned char base_start_rcv=0;
unsigned char TCCR2_cal=0;
unsigned int base;
unsigned int baud=9600;
void soft_serout_init(char tx,unsigned long baud_)
// Конфигурирование и запуск таймера timer2
{
unsigned long tick=0;
out_c(tx,1);
if(baud_<=4800)
{
tick = ((1000000/baud_)-OFFSET_DELAY1)/4; // Вычисление задержки для
                                          // заданной скорости обмена
                                          // данными
TCCR2_cal = PRESCALER_64;
}
else if(baud_>4800 && baud_<=9600)
{
tick = ((1000000/baud_)-OFFSET_DELAY2)/2; // Вычисление задержки для
                                          // заданной скорости обмена
```

```
                                        // данными

TCCR2_cal = PRESCALER_32;
}
TCCR2 = 0; // Остановка таймера
TIFR |= 1<<TOV2; // Гарантированная очистка флага прерывания
base = 255-tick;
base_start_rcv = 255-(tick/2);
}
// Delay for baudrate
void delay_baud(unsigned int _tick)
{
TCNT2 = _tick; // Загрузка в предварительный делитель вычисленного
TCCR2 = TCCR2_cal; // Загрузка значения интервала
while(!(TIFR & (1<<TOV2))); // Ожидание окончания счета
TIFR |= 1<<TOV2; // Гарантированная очистка флага переполнения
TCCR2 = 0; // Остановка таймера Timer2
}
// Передача одного байта данных
void serout_byte(char tx,unsigned char dat)
{
int i;
out_c(tx,0); // стартовый бит
delay_baud(base); // Задержка для формирования стартового бита
for(i=0;i<8;i++)
{
out_c(tx,dat & 0x01); // Передача одного бита данных
delay_baud(base); // Задержка для заданной скорости обмена
dat=dat>>1; // Сдвиг для получения следующего бита
}
out_c(tx,1); // Стоповый бит
delay_baud(base); // Задержка для формирования стопового бита
}
// Функция передачи более чем одного байта
void serout_text(char tx,unsigned char *p)
{
while(*p)
{ serout_byte(tx,*p++); }
}
#endif
```

COPYRIGHTS