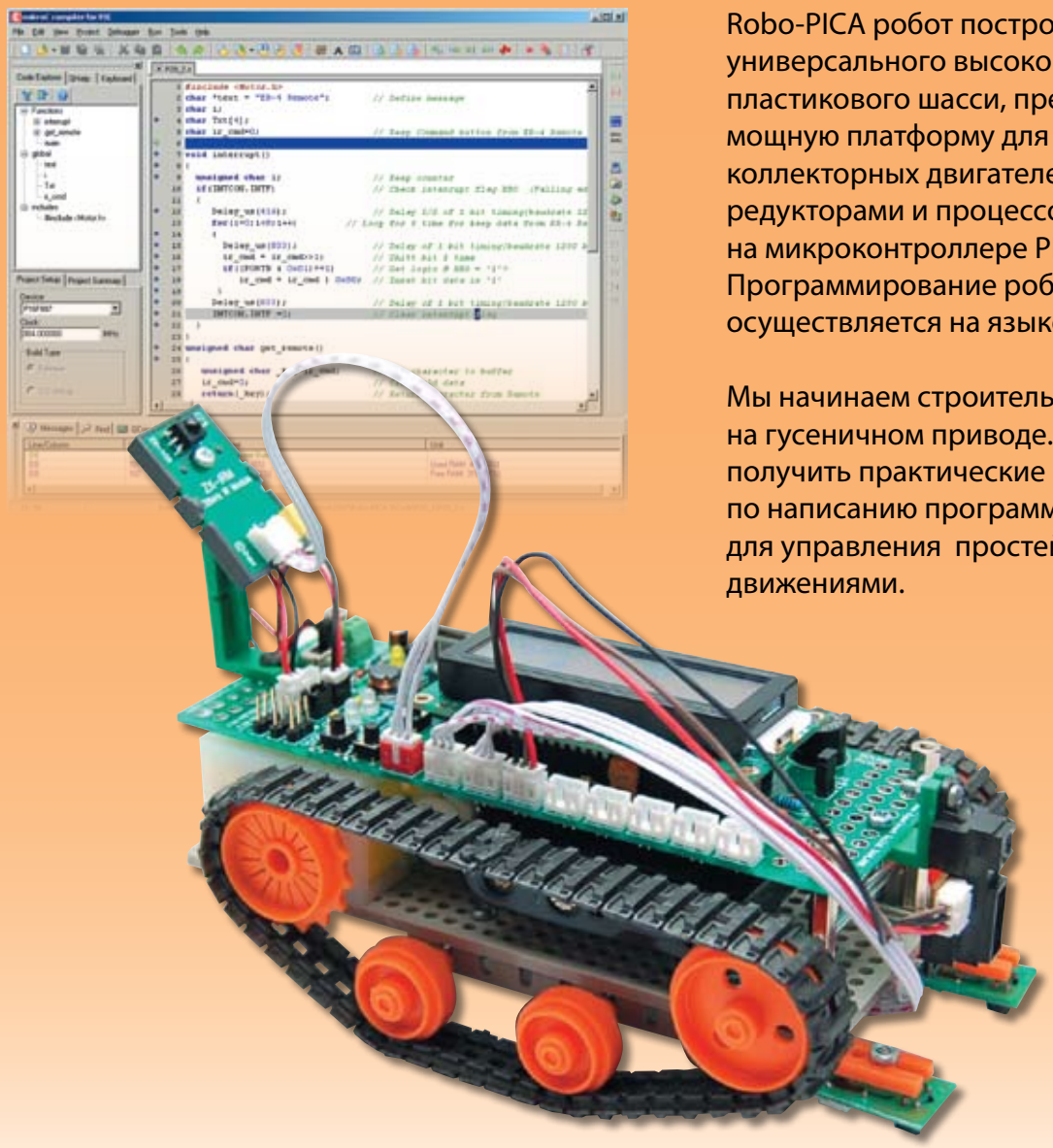
ROBO PICA

PIC MICRO.ROBOT KIT

- Программируй микро-робот на PIC микроконтроллере
- Изучай программирование микроконтроллеров на Си
- Развлекайся, исследуя роботов

СОЗДАЙ СВОЕГО ПЕРЕПРОГРАММИРУЕМОГО МОБИЛЬНОГО РОБОТА НА PIC-КОНТРОЛЛЕРЕ



Robo-PICA робот построен на базе универсального высококачественного пластикового шасси, представляющего мощную платформу для установки коллекторных двигателей с редукторами и процессорной платы на микроконтроллере PIC16F887. Программирование робота осуществляется на языке Си.

Мы начинаем строительство робота на гусеничном приводе. Это позволит получить практические навыки по написанию программного кода для управления простейшими движениями.

Инструкция по сборке и программированию

www.inexglobal.com

inex INNOVATIVE EXPERIMENT

МЫ СОКРАЩАЕМ ВРЕМЯ РАЗРАБОТКИ!

СОВРЕМЕННЫЕ ЭЛЕКТРОННЫЕ КОМПОНЕНТЫ → **ОТ 1 ШТУКИ СО СКЛАДА**

СРЕДСТВА РАЗРАБОТКИ И ОТЛАДКИ → **БОЛЕЕ 800 НАИМЕНОВАНИЙ НА СКЛАДЕ + АРЕНДА**

ЭЛЕКТРОННЫЙ КАТАЛОГ
(www.terraelectronica.ru) → **ПАРАМЕТРИЧЕСКИЙ ПОИСК ПО СРЕДСТВАМ РАЗРАБОТКИ И ЭК**

КОНСУЛЬТАЦИИ СПЕЦИАЛИСТОВ → **ПО ВЫБОРУ И ПРИМЕНЕНИЮ СРЕДСТВ РАЗРАБОТКИ**

ЗАКАЗ ПЕЧАТНЫХ ПЛАТ → **ОТ 1 ШТУКИ ЛЮБОГО РАЗМЕРА ОТ 1 ДО 64 СЛОЕВ**

www.terraelectronica.ru

Робототехнический эксперимент с PIC-микроконтроллером,

основанный на демонстрационном наборе Robo-PICA

3-е Издание

Оглавление

microC – зарегистрированная торговая марка фирмы microElektronika (www.mikroe.com).

PIC и PICkit2™ – зарегистрированная торговая марка фирмы Microchip Technology (www.mikrochip.com).

Глава 1

Описание деталей Robo-PICA и Введение в программное обеспечение

1.1 Описание деталей Robo-PICA

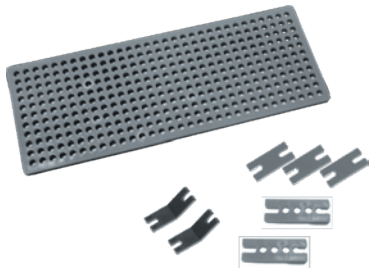
Описание состоит из 2 частей:
1.1.1 Описание механических комплектующих
1.1.2 Описание электронных комплектующих

1.1.1 Механические комплектующие

Блок двигателя с редуктором – состоит из корпуса с элементами крепления; двигателя постоянного тока, с напряжением питания 4,5 В (максимум 9 В) и потребляемым током 180 мА; редуктора, с передаточным отношением 48:1 и максимальным моментом 4 кг/см.

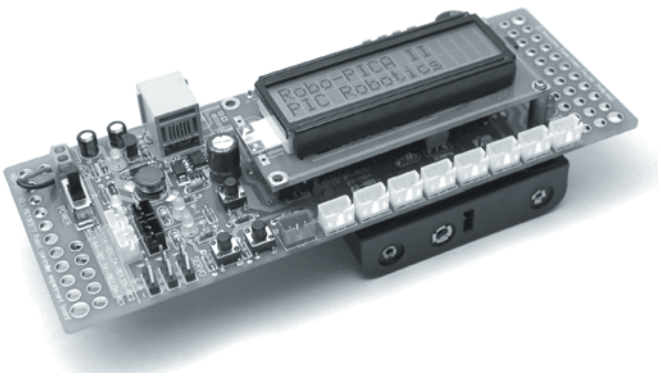
Набор гусениц – включает в себя гусеницы трех типоразмеров, совместимые с множеством типов колес и гусеничных лент, осей и оснований

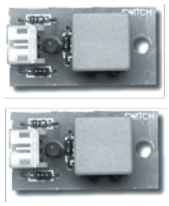
Винты нескольких размеров и гайки
(Винты: 3x6 мм,3x10 мм, 3x15 мм, 3x25 мм и 3x35 мм, 3 мм гайки), Винты с плоскими и закругленными головками.
Набор пластиковых втулок
(длина: 3 мм, 15 мм и 25 мм)
Шестигранные стойки:
3x30 мм.



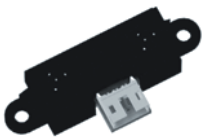
Монтажная плата и 4 вида различных цветных пластиковых крепежных пластин
(10 плоских пластин,
10 прямоугольных пластин,
10 тупоугольных пластин и
плоские пластины
с 3/5/12 отверстиями)

1.1.2 Электронные комплектующие





ZX-01
Концевые выключатели (2шт.)

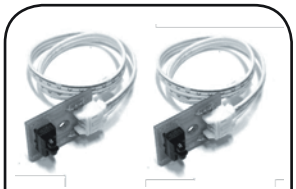


GP2D120
Инфракрасный измеритель расстояния с дальностью 4-30 см

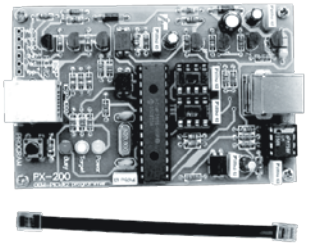
RBX-877V2.0 PIC16F887 Плата для робототехнических экспериментов

ZX-IRM
Инфракрасный приемник на частоту 38 кГц

ER-4
Инфракрасный пульт дистанционного управления



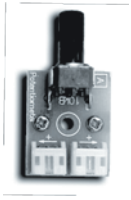
ZX-03
Инфракрасный отражатель (2 шт.)



Плата USB-программатора с ICD2 кабелем

USB-кабель

4 батареи AA (Рекомендуется использовать перезаряжаемые аккумуляторные батареи – они не включены в набор)



ZX-POTH
Потенциометр (1шт.)

1.2 Инструменты для сборки демонстрационного набора

Кусачки

Нож для бумаги со сменными лезвиями

Отвертка Филипс

Компьютер, с установленной ОС Windows 98 или выше, имеющий как последовательный (RS232), так и параллельный порт.

1.3 Программные средства разработки для программирования Робота

Набор RoboPICA использует PIC-микроконтроллер PIC16F887. Конструктор может написать управляющую программу на языках Ассемблер, BASIC или C. Только для программ на языках BASIC или C необходимо использование специальной программы - компилятора.

Все примеры для этого набора написаны на языке C для компилятора mikroC компании mikroElektronika (mikroE: www.mikroe.com). Набор для создания робота Robo-PICA также может использовать этот компилятор.

При создании программ набора для создания робота используется демо-версия компилятора Mikro C. Разработчикам, которым необходимо создавать более сложные версии управляющих микропрограмм, необходимо будет купить полную версию компилятора либо на веб-сайте компании MikroElektronika либо у ее официальных партнеров. Демо-версию компилятора mikroC можно загрузить непосредственно с веб-сайта http://www.mikroe.com. Однако, в наборе для создания робота Robo-PICA, это программное обеспечение записано на входящий в набор CD-ROM. Вам будет необходимо загрузить с веб-сайта компании mikroElektronika только последнюю версию руководства пользователя для этого компилятора. В имеющемся на диске руководстве может не быть описания некоторых функций.

Другим инструментом, необходимым для работы с PIC-микроконтроллером, является программное обеспечение для программирования микроконтроллера. В состав набора Robo-PICA включен USB-программатор. По своим функциям он совместим с программатором PICkit2™ компании Microchip. Компиляторы mikroE могут использовать программное обеспечение PICkit2™. Загрузить программное обеспечение PICkit2™ можно с сайта www.microchip.com.

1.3.1 Компилятор mikroC (Демо-версия)

1.3.1.1 Краткий обзор

mikroC является мощным, обладающим богатыми функциональными возможностями инструментом для разработки устройств на базе PIC-микроконтроллеров. Он создан, чтобы предоставить пользователю максимально удобное средство создания решений для встраиваемых систем, обладает высокой производительностью и прост в использовании.

mikroC предоставляет разработчику полнофункциональную, обладающую множеством передовых возможностей, графическую оболочку (IDE), совместимый со стандартом ANSI компилятор, большой набор библиотек для работы с аппаратными модулями, подробную документацию и множество готовых к использованию примеров.

mikroC позволяет быстро создавать и отлаживать сложные приложения:

- Писать собственные исходные тексты на языке C, используя многофункциональный Редактор Кода.

Особые благодарности: за всю информацию о Компиляторе mikroC и Программаторе PICkit2, полученную с веб-сайтов их производителей (www.mikroe.com и www.microchip.com) и технической документации. Спасибо за все свободно распространяемые средства разработки с открытыми исходными кодами. Пользователи, которым необходим полнофункциональный компилятор mikroC, могут купить его непосредственно на сайте www.mikroe.com.

- Резко увеличивать скорость разработки, используя встроенные библиотеки функций mikroC: обработки данных, работы с памятью, отображения информации, различных преобразований, обмена данными...
- Отслеживать структуру программы, переменные и функции в Code Explorer. Создавать удобный для восприятия ассемблерный код с подробными комментариями и стандартные HEX-файлы, совместимые со всеми программаторами.
- Исследовать процесс выполнения программы и отлаживать логику ее работы с использованием встроенного Отладчика. Получать детальные отчеты и графы статистики программного кода, ассемблерный текст, дерево вызовов процедур и функций и многое другое...
- mikroE предлагает законченные примеры для расширения Ваших знаний, опыта разработки и использования в качестве кирпичиков в Ваших собственных проектах.
- Демоверсия является полнофункциональной средой разработки и имеет единственное ограничение: размер выходного HEX-файла не может превышать 2к программных слов.

1.3.1.2 Установка демоверсии компилятора mikroC

Загрузите последнюю версию компилятора с вебсайта компании mikroElektronika; www.mikroe.com. Запустите установочный файл. Дополнительно следует загрузить 5 файлов документации:

- "Руководство по mikroC" ("mikroC manual"),
- "Создание первого проекта на языке mikroC для PIC-микроконтроллеров" ("Creating First Project in mikroC for PIC"),
- "Справочник по языку C" ("Reference Guide for C language"),
- "Графическая оболочка компилятора" ("Compilers IDE document")
- "Получение и Активация Лицензионного Ключа" ("Obtaining and Activating the License Key").

В документе "Руководство по mikroC" описан полный синтаксис языка C и детально описаны все библиотечные функции. В текущем руководстве описаны только те особенности языка программирования, которые необходимы для успешного выполнения лабораторных работ, посвященных функционированию собираемого из набора робота.

1.3.2 Программное обеспечение программатора PICkit2™

Оболочка для Программирования микроконтроллеров PICkit2™ позволяет программировать большинство Flash-микроконтроллеров компании Microchip. Для уточнения возможности программирования специфических микроконтроллеров следует обратиться к файлу README, или проверить на веб-сайте компании Microchip.

Полнофункциональный интерфейс программирования для ОС Windows поддерживают PIC-микроконтроллеры основных семейств (PIC10F, PIC12F5xx, PIC16F5xx), микроконтроллеры среднего уровня (PIC12F6xx, PIC16F), PIC18F, PIC24, dsPIC30 и dsPIC33 семейства 8-битных и 16-битных микроконтроллеров и многие микросхемы последовательных запоминающих устройств EEPROM компании Microchip.

Оболочка для программирования микроконтроллеров PICkit2™ работает с OEM USB-программатором PICkit2™. USB-программатор функционирует как внутрисхемный программатор и подключается через разъем ICD2.

1.3.2.1 Установка Программного обеспечения PICkit2™

1.3.2.1.1 Установка с компакт диска PX-200

Рабочим программным обеспечением для USB-программатора является Оболочка Программирования PICkit2™. Последние ее версии созданы с использованием Microsoft.NET. Поэтому перед установкой PICkit2™ необходимо установить Microsoft.NET Framework.

(А) Установка пакета Microsoft .NET Framework

Первым необходимым действием является установка Microsoft.NET Framework. Найдите на CD-ROM каталог **PICkit 2 Setup v2.01 dotNET → dotnetfx**. Запустите двойным щелчком по левой кнопке мыши файл **dotnetfx.exe**. После полной установки Microsoft.NET Framework, установите Оболочку Программирования PICkit2™ двойным щелчком по левой кнопке мыши на файле **PICkit2Setup.msi**. При этом запустится установщик программного обеспечения.

(Б) Пакет Microsoft .NET Framework уже был установлен ранее

При этом нет необходимости в повторной установке Microsoft.NET Framework. Можно сразу устанавливать Оболочку Программирования PICkit2™, перейдя в каталог **PICkit 2 Setup v2.01x** на входящем в комплект набора Robo-PICA CD-ROM. После двойного щелчка левой кнопкой мыши на файле **PICkit2Setup.msi** начнется установка программного обеспечения.

1.3.2.1.2 Установка из Интернет.

Посетите веб-сайт компании Microchip ***www.microchip.com***. Перейдите по ссылке ***Development tools*** и затем по ссылке ***PICkit 2 Programmer/Debugger***.

(А) Установка пакета Microsoft .NET Framework

Пользователям, у которых не установлен пакет Microsoft .NET Framework, необходимо вначале установить его, загрузив необходимый файл из раздела ***PICkit2V2.01 Install with .NET Framework***. После загрузки файла **PICkit 2 Setup v2.01 dotNET.zip** (номер версии может отличаться от указанного), распакуйте его в каталог **PICkit 2 Setup v2.01 dotNET**. После этого перейдите в него, а затем в подкаталог **dotnetfx**. Запустите двойным щелчком по левой кнопке мыши на файле **dotnetfx.exe** инсталляцию Microsoft .NET Framework. После полной установки Microsoft.NET Framework, установите Оболочку Программирования PICkit2™ двойным щелчком по левой кнопке мыши на файле **PICkit2Setup.msi**. При этом запустится установщик программного обеспечения.

(Б) Пакет Microsoft .NET Framework уже был установлен ранее

Пользователям, у которых уже установлен пакет Microsoft .NET Framework, рекомендуется загружать установочный файл из раздела ***PICkit2V2.01 Install***. Необходимо загрузить файл **PICkit 2 Setup v2.01.zip** (номер версии может отличаться от указанного). После распаковки загруженного файла в каталог **PICkit 2 Setup v2.01**, необходимо перейти в него, и двойным щелчком левой кнопкой мыши запустить файл **PICkit2Setup.msi**, после чего начнется процесс установки.

После запуска установочного файла; **PICkit2Setup.msi**. кликайте по кнопке <Принять> (<Accept>) на каждом шаге инсталляции до тех пор, пока процесс установки не завершится полностью.

1.3.2.2 Использование Оболочки Программатора PICkit2™

1.3.2.2.1 Проверка правильности подключения программатора

1. Соедините USB-кабелем программатор и USB-порт компьютера. Запустите Оболочку Программатора Pickit2™, нажав кнопку <Пуск> (<Start>) системного меню и выбрав из него пункты Программы(All programs) → Microchip → Pickit 2 V201. После этого появится Главное окно, внешний вид которого показан на рис. 1-1.
2. При правильном соединении в поле Состояния появится текст **PICkit 2 found and connected** (**PICkit 2 найден и с ним установлено соединение**).

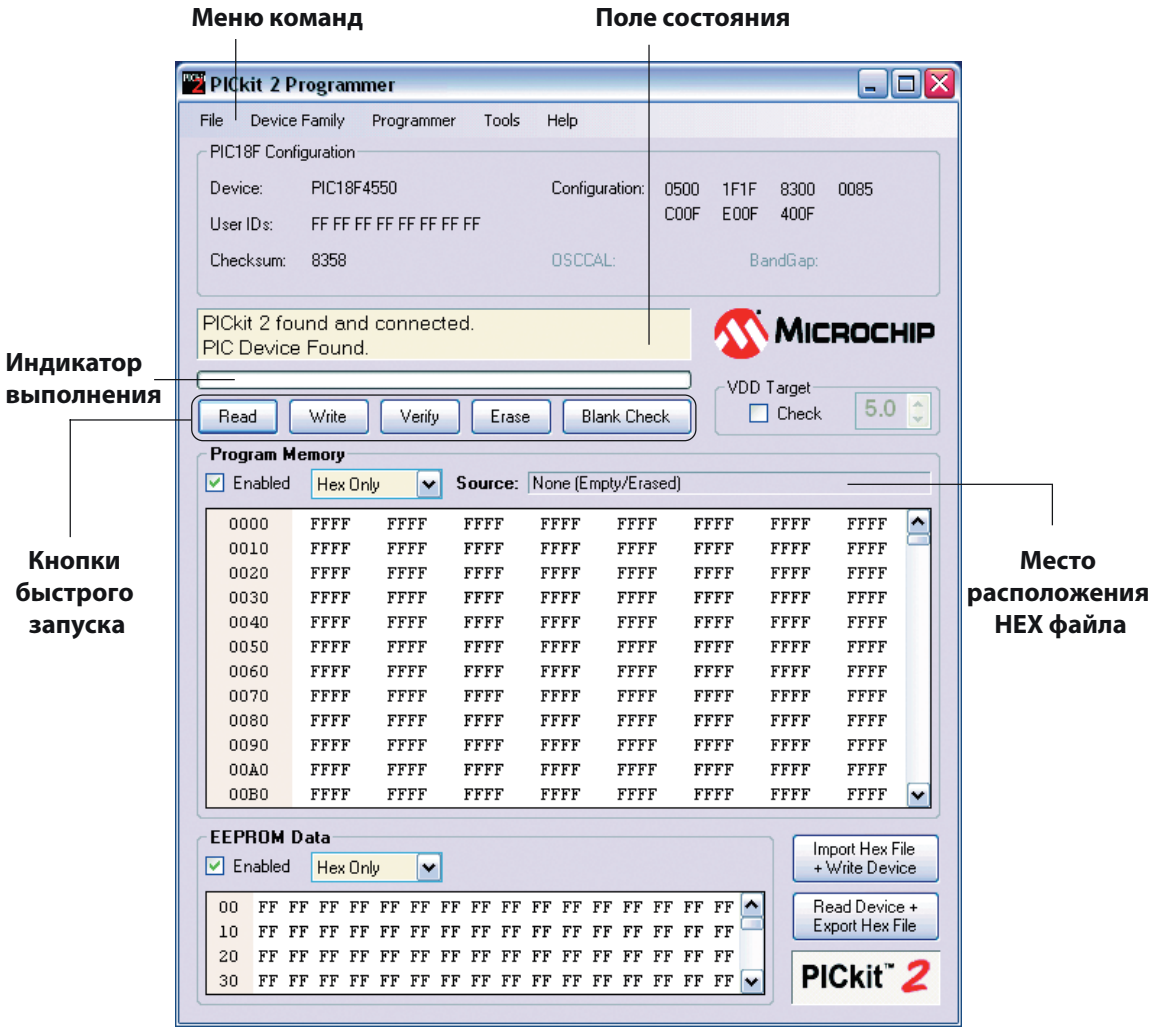
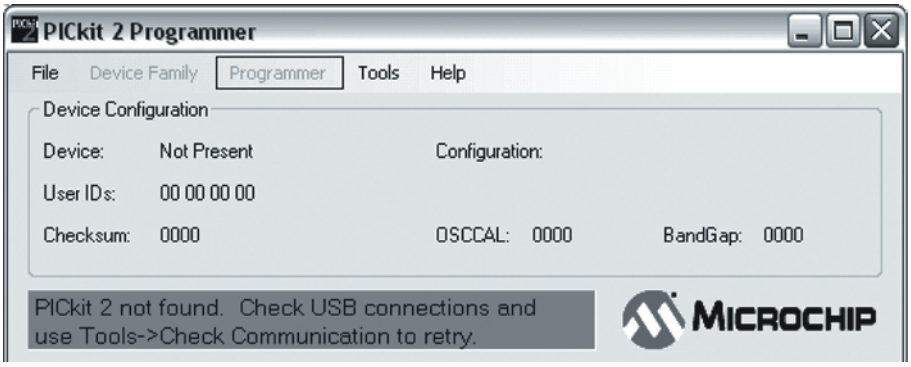
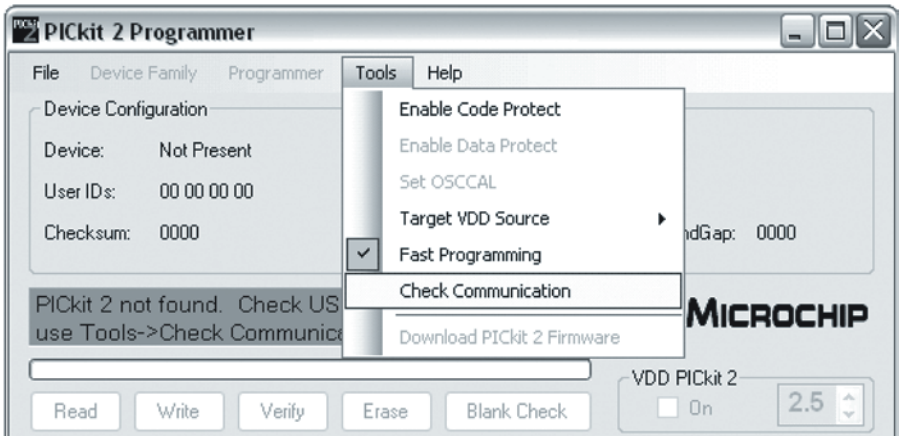


Рис. 1-1. Главное окно оболочки PicKit2™

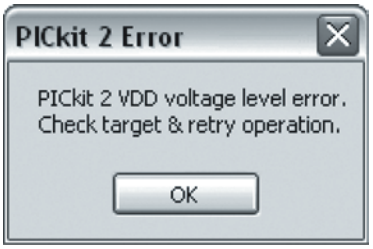
3. При неправильном соединении. В поле Состояния появится сообщение **PICkit 2 not found. Check USB connections and use Tools → Check Communication to retry (PICkit 2 не найден. Проверьте USB-кабель и повторите попытку, используя пункты меню Tools (Инструменты) → Check Communication (Проверка Соединения))**. Проверьте наличие кабеля и надежность соединения (кабель должен быть плотно вставлен в разъемы как программатора, так и компьютера).



4. Перейдите в меню **Tools (Инструменты)** и выберите команду **Check Communication (Проверка Соединения)**. Если все заработало правильно, в поле Состояния появится сообщение **PICkit 2 found and connected** (**PICkit 2 найден и с ним установлено соединение**).



Однако, каждый раз во время переподключения или проверки аппаратных средств, при отсутствии подключения программируемого микроконтроллера к разъему ICD2 и выводу ICSP или несоответствии в нумерации выводов ИС, будет выведено следующее диалоговое окно с предупреждением. Оно будет сообщать о неправильном напряжении питания. Не беспокойтесь об этом и для продолжения нажмите кнопку <**OK**>.



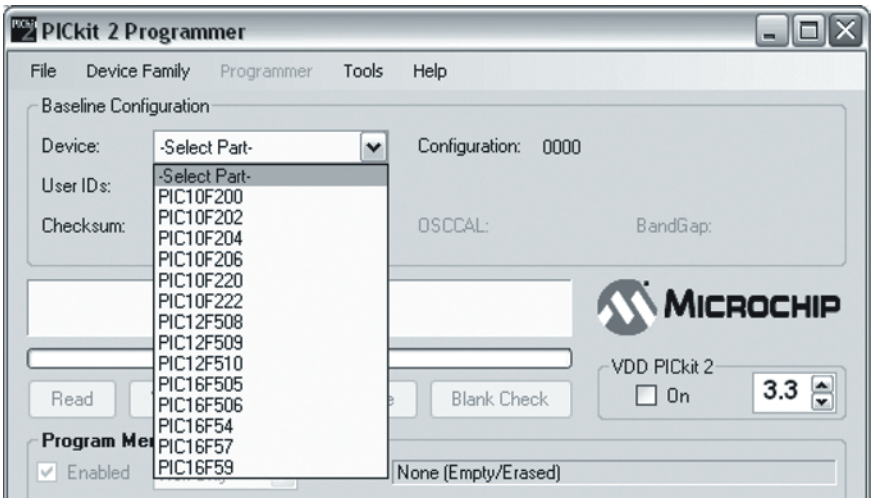
1.3.2.2.2 Описание меню команд

FILE (ФАЙЛ)

- **Import File (Импортировать файл)** – Импортировать HEX-файл для программирования
- **Export File (Экспортировать файл)** – Экспортировать HEX-файл, прочтенный из микросхемы
- **Exit (Выход)** – Завершить работу программы (дублируется кнопкой **<Quit>**)

DEVICE FAMILY (СЕМЕЙСТВО МИКРОСХЕМ)

- **Baseline (Основное семейство)** (12-битное ядро) – Настраивает Оболочку для Программирования для работы с основным семейством Flash-микросхем.



- **Mid-range (Средний уровень)** – Настраивает Оболочку для Программирования для работы с flash-микросхемами с 14-битным ядром. В список включены микросхемы семейств PIC12F6xx and 16F6xx, 7x, 7xx, 8x, 8xx. После выбора микросхемы, программное обеспечение будет проверять подключение программируемой микросхемы к разъему ICD2 и контакту ICSP. При обнаружении выбранной микросхемы, ее обозначение появится в строке **Device** в панели **Midrange Configuration**. Для продолжения нажмите кнопку **<OK>**. Для платы RBX-877V2.00 необходимо использовать эту группу микросхем, поскольку на плате установлена управляющая микросхема PIC16F887, относящаяся к данной группе PIC-микроконтроллеров.
- **PIC18F** – Настраивает Оболочку для Программирования для работы с flash-микросхемами с ядром PIC18F.
- **PIC18F_J_** – Настраивает Оболочку для Программирования для работы с семейством PIC18FxxJxx микросхем с низким напряжением питания.
- **PIC24** - Настраивает Оболочку для Программирования для работы с микросхемами семейства PIC24FJxx с 16-битным ядром.
- **dsPIC30** – Настраивает Оболочку для Программирования для работы с микросхемами семейства dsPIC30Fxx с 16-битным ядром.
- **dsPIC33** – Настраивает Оболочку для Программирования для работы с микросхемами семейства dsPIC33Fxx с 16-битным ядром.

PROGRAMMER (ПРОГРАММАТОР)

- **Read Device (Чтение микросхемы)** – Считывает память микропрограмм, данные EEPROM памяти, область ID и Конфигурационных битов.
- **Write Device (Запись микросхемы)** – Записывает данные в память микропрограмм, EEPROM память, область ID locations и Конфигурационные биты.
- **Verify (Проверка)** – Проверяет память микропрограмм, данные EEPROM памяти, область ID и Конфигурационные биты, сравнивая данные, прочитанные из микросхемы с данными, запомненными в Оболочке для Программирования.
- **Erase (Стирание)** – Выполняет полное стирание программируемой микросхемы. Значения OSCCAL (калибровка внутреннего RC-генератора) и band gap (ширина запрещенной зоны) остаются неизменными (только для PIC12F629/675 и PIC16F630/676).
- **Blank Check (Проверка стирания)** – Выполняет проверку стирания памяти микропрограмм, данных EEPROM памяти, области ID и Конфигурационных битов.
- **Verify on Write (Проверка после Записи)** – Осуществляет проверку памяти микропрограмм, данных EEPROM памяти, области ID и Конфигурационных битов, считанных после выполнения операции программирования из программируемой микросхемы с данными, запомненными в Оболочке для Программирования, слово в слово.
- **Full Erase (OSCCAL and BG erased) (Полное стирание, включая стирание значений OSCCAL и BG)** – Выполняет полное стирание микросхемы, включая значения OSCCAL и Band Gap (BG) (только для PIC12F629/675 и PIC16F630/676).
- **Regenerate OSCCAL (Восстановление значения OSCCAL)** – Восстанавливает значение OSCCAL (только для PIC12F629/ 675 и PIC16F630/676). Линию AUX необходимо подсоединить к выводу RA4/T1G.
- **Set Band Gap Calibration Value (Установка калибровочного значения для BG)** – Устанавливает значение band gap (ширины запрещенной зоны).
- **Write on PICkit Button (Запись по нажатию на кнопку в PICkit)** – Разрешает поддержку программирования микроконтроллера по нажатию на кнопку <PROGRAM> на плате USB-программатора.

TOOLS (ИНСТРУМЕНТЫ)

- **Enable Code Protect (Разрешить защиту кода)** – Разрешает защиту кода для Flash-памяти микропрограмм.
- **Enable Data Protect (Разрешить защиту данных)** – Разрешает защиту кода для EEPROM-памяти данных.
- **Set OSCCAL (Установить значение OSCCAL)** – Устанавливает значение OSCCAL для точной настройки частоты внутреннего тактового RC-генератора.
- **Target VDD Source (Источник напряжения питания программируемой системы)** – Устанавливает питание программируемой системы от USB-программатора.

Auto-Detect (Авто-обнаружение): Разрешает USB-программатору автоматически подавать напряжение питания на программируемый микроконтроллер и автоматически снимать его (начинающим не рекомендуется использовать этот режим).

Forced PICkit2 (Принудительно PICkit2): Режим использования программатора для подачи заданного напряжения питания на программируемый микроконтроллер. После выбора этого режима загорается светодиод "Target", и флажок VDD PICkit2 на экране устанавливается в положение On (выбрано). Пользователь может изменять напряжение питания в окошке выбора с правой стороны окна программы (начинающим не рекомендуется использовать этот режим).

Forced Target (Принудительно программируемая система): При выборе этого пункта, программное обеспечение само определяет величину питающего напряжения для программируемой системы. Рекомендуется использовать этот режим для безопасного программирования. Кроме того, в этом пункте пользователь должен определить величину напряжения питания программируемого PIC-микроконтроллера.

- **Fast Programming (Быстрое Программирование)** – Выбор PX-200 для программирования Flash-микросхем с высокой скоростью.
- **Check Communication (Проверка подключения)** – Проверяет обмен данными между компьютером и USB-программатором и считывает значение Device ID программируемой микросхемы.
- **Download PICkit 2 Firmware (Загрузка Микропрограммы в PICkit 2)** – Выполняет загрузку микропрограммы операционной системы USB-программатора. (Входящий в набор USB-программатор аналогичен Программатору PICkit2™). Эта функция вызывается периодически для обновления ОС-программатора.

Help (Помощь)

Вызывает для просмотра все руководства пользователя, техническую документацию и диалоговое окно "О программе", показывающее версию и дату создания программы.

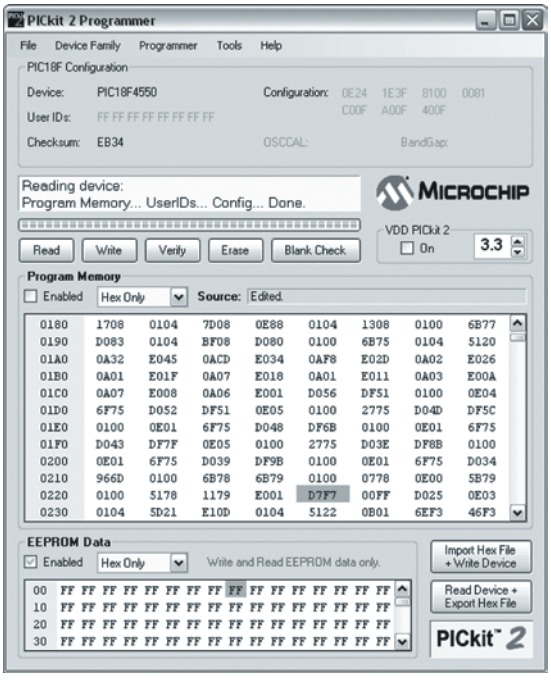
1.3.2.2.3 Важная информация, которую необходимо знать при использовании Оболочки для Программирования PICkit2™.

Editing memory value (Редактирование ячеек памяти)

Оболочка для Программирования PICkit2™ позволяет редактировать значения отдельных ячеек памяти как Flash-памяти микропрограмм, так и EEPROM памяти данных. Пользователь может выбрать любой адрес для изменения значения записанных по нему данных и непосредственно ввести новое значение. Кроме того, пользователь может выбрать, будет ли осуществляться доступ к обоим типам памяти или только к одному.

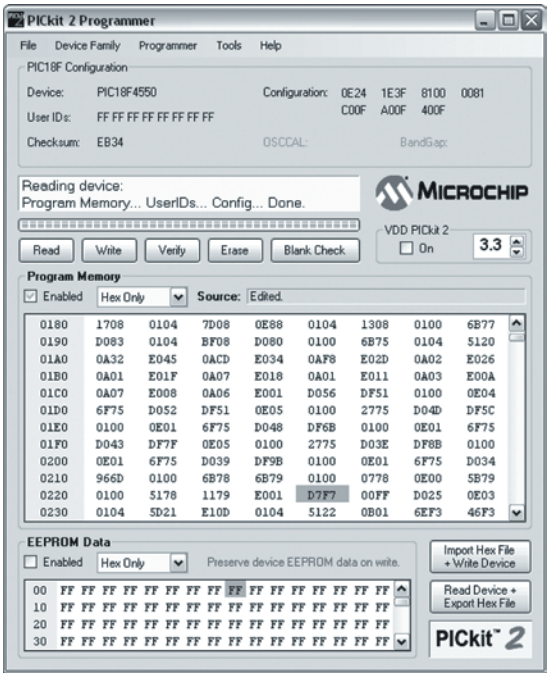
(a)Access only EEPROM data memory (Доступ только к EEPROM памяти данных)

Снимите отметку с флажка *Enabled* (*Разрешить*) в рамке группы *Program Memory (Память микропрограмм)*. После этого в рамке группы *EEPROM Data (EEPROM данных)* появится сообщение ***WRITE AND READ EEPROM DATA ONLY (ЗАПИСЬ И ЧТЕНИЕ ТОЛЬКО EEPROM-ДАННЫХ)*** красного цвета. Это означает, что можно выполнять операции Записи и Чтения только с EEPROM данных. См. приведенную ниже иллюстрацию:



(б) Access only Flash program memory (Доступ только к Flash-памяти микропрограмм)

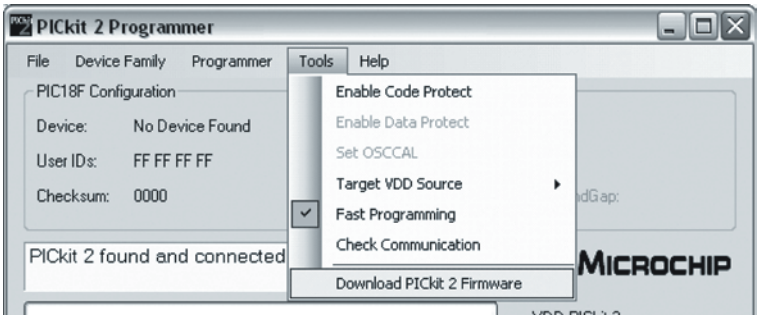
Снимите отметку с флажка ***Enabled*** (***Разрешить***) в рамке группы ***EEPROM Data (EEPROM данных),*** при этом появится сообщение ***PRESERVE DEVICE EEPROM DATA ON WRITE (ЗАЩИТА ДАННЫХ EEPROM ОТ ЗАПИСИ)*** красного цвета. Это означает, что EEPROM память данных будет защищена. Пользователь будет иметь доступ только к Flash-памяти микропрограмм. См. приведенную ниже иллюстрацию:



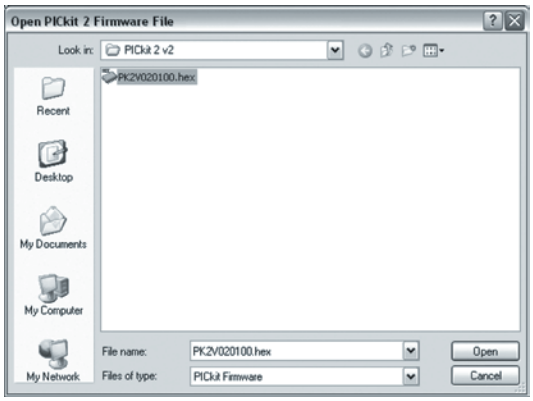
1.3.2.3 Обновление микропрограммы USB-программатора

Чтобы обновить микропрограмму Операционной Системы программатора полностью, выполните следующую последовательность операций:

1. Загрузите последнюю Операционную Систему PICkit 2™ с веб-сайта компании Microchip www.microchip.com. Использование указанной ОС допустимо, поскольку USB-программатор Robo-PICA аналогичен программатору PICkit2™ компании Microchip.
2. Из меню выберите **Tools → Download PICKit 2 OS Firmware**, как показано на рисунке ниже



3. Выберите для просмотра каталог, в который Вы сохранили файл последней Операционной Системы. Отметьте файл **PK2*.hex** и нажмите кнопку **Open (Открыть)** как показано на рисунке ниже:



4. Ход выполнения операции обновления ОС будет отображаться в строке состояния Оболочки для программирования и светодиод Busy (Ожидание) LED на USB-программаторе Микроконтроллеров будет при этом мигать. После полного завершения обновления, в строке состояния появится сообщение **"Operating System Verified" ("Проверка загрузки Операционной Системы")** и светодиод Busy погаснет. После этого обновление операционной системы можно считать завершенным.

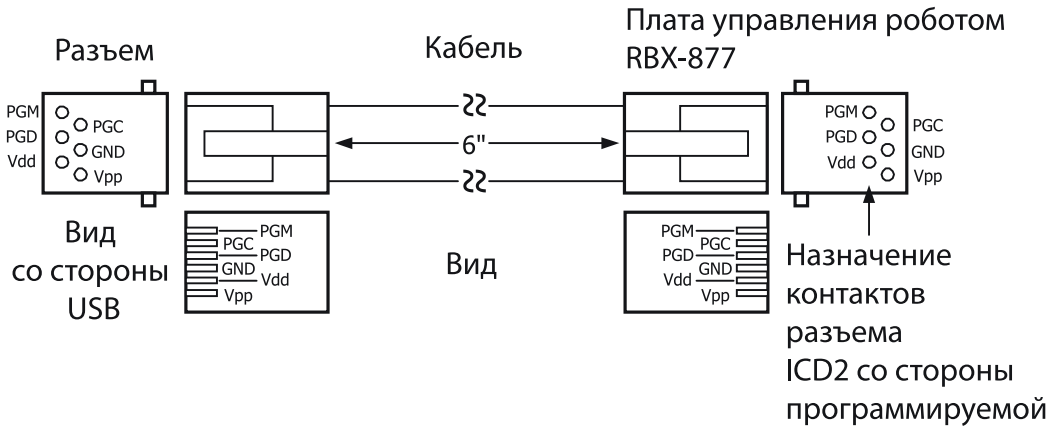
1.3.2.4 Short cut button (Горячие клавиши)

Программное обеспечение PICkit2™ имеет следующие 7 горячих клавиш:

1. **Read (Чтение):** чтение данных из программируемой микросхемы.
2. **Write (Запись):** запись или программирование кода в микросхему.
3. **Verify (Проверка):** проверка программирования.
4. **Erase (Стирание):** стирание данных в программируемой микросхеме.
5. **Blank Check (Проверка стирания):** проверка очистки (стирания) программируемой микросхемы.
6. **Import HEX File + Write Device (Импорт HEX-файла и Программирование Микросхемы):** Открыть HEX-файл и автоматически записать его в программируемую микросхему
7. **Read Device + Export HEX File (Чтение Микросхемы и Экспорт HEX-файла):** прочитать данные из микросхемы и автоматически записать их в HEX-файл.

1.3.2.5 Назначение контактов ICD2-кабеля

USB-программатор комплектуется ICD2-кабелем для передачи данных между программатором и программируемым устройством. Назначение контактов этого кабеля показано ниже:



1.4 Разработка программы управления для Robo-PICA

Общая схема создания программы управления для робота, собираемого из набора Robo-PICA, выглядит следующим образом:

1. Создание файла проекта на языке C с использованием программы mikroC IDE.
2. Компиляция файла проекта (при этом текст проекта проверяется на наличие синтаксических ошибок и, в случае их отсутствия, создается выходной HEX-файл).
3. При обнаружении ошибок – редактирование программы на языке C и компилирование проекта до тех пор, пока все ошибки не будут исправлены.
4. После успешной компиляции, т.е., если в проекте отсутствуют синтаксические ошибки, создается выходной HEX-файл.
5. Запустить программное обеспечение PICkit2™-программатора. Соединить кабелем USB-порт компьютера с USB-программатором и соединить ICD2-кабелем USB-программатор плату управления RBX-877 V2.0 через разъемы ICD2.
6. Загрузить HEX-файл в плату Управления RBX-877 V2.0 Робота-PICA.
7. Запустить программу на выполнение и проверить выполнение Роботом всех необходимых операций. При обнаружении любых ошибок в работе робота, вернуться к редактированию программы на языке C, скомпилировать и еще раз загрузить полученный HEX-файл в плату Управления. Проделать эту последовательность действий до тех пор, пока робот не будет правильно выполнять все операции.

1.5 Начало работы с набором Robo-PICA

Начиная с этого момента, будет дано введение в разработку программного обеспечения (микропрограмм, firmware) для набора Robo-PICA. Робот, создаваемый из этого набора, будет управляться платой Управления Роботом RBX-877 V2.0. Сердцем этого контроллера является микросхема PIC16F887. Разработка программного обеспечения состоит из 2 основных этапов: Разработка Программы Управления на языке C и Загрузка созданного на предыдущем шаге HEX-файла в микроконтроллер.

Для разработки программы управления будет использоваться язык C и графическая среда разработки mikroC IDE, включающая компилятор языка C и другие полезные инструменты разработки, и готовые библиотеки функций. Однако, следует отметить, что этот набор комплектуется демо-версией mikroC IDE. При необходимости, полную версию mikroC IDE можно купить на веб-сайте **www.mikroe.com** или у партнеров компании Mikroelektronika.

Демо-версия позволяет создавать проекты на языке C и тестировать правильность работы аппаратной платформы Robo-PICA при помощи описанной ниже процедуры:

1.5.1 Следуя инструкциям руководства, установите среду разработки mikroC. Подробности установки можно найти в документе на компакт диске, включенном в набор Robo-PICA или загрузить обновленную версию с веб-сайта www.mikroe.com.

1.5.2 Установите Программное обеспечение PICkit2 для USB-программатора.

1.5.3 Откройте mikroC IDE, выбрав в системном меню Start(Пуск) → Programs (Программы) → Mikroelktronika → mikroC → mikroC. После этого на экране появится главное окно mikroC IDE. На рис. 1-1 показано главное окно mikroC IDE и его важные компоненты.

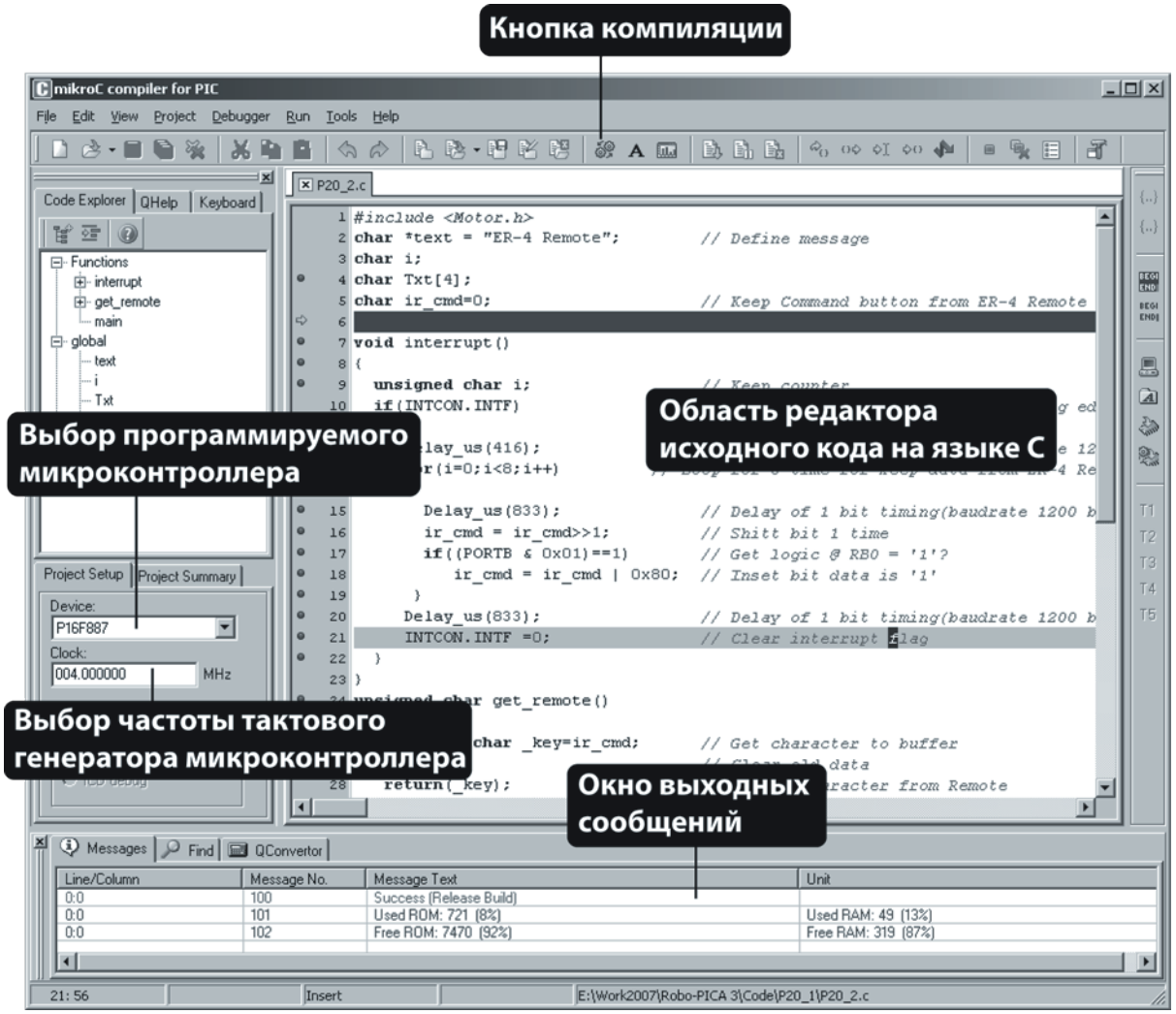
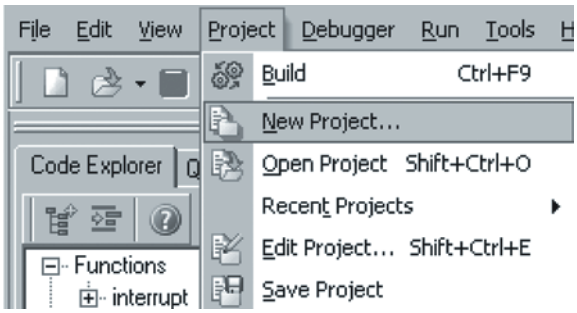
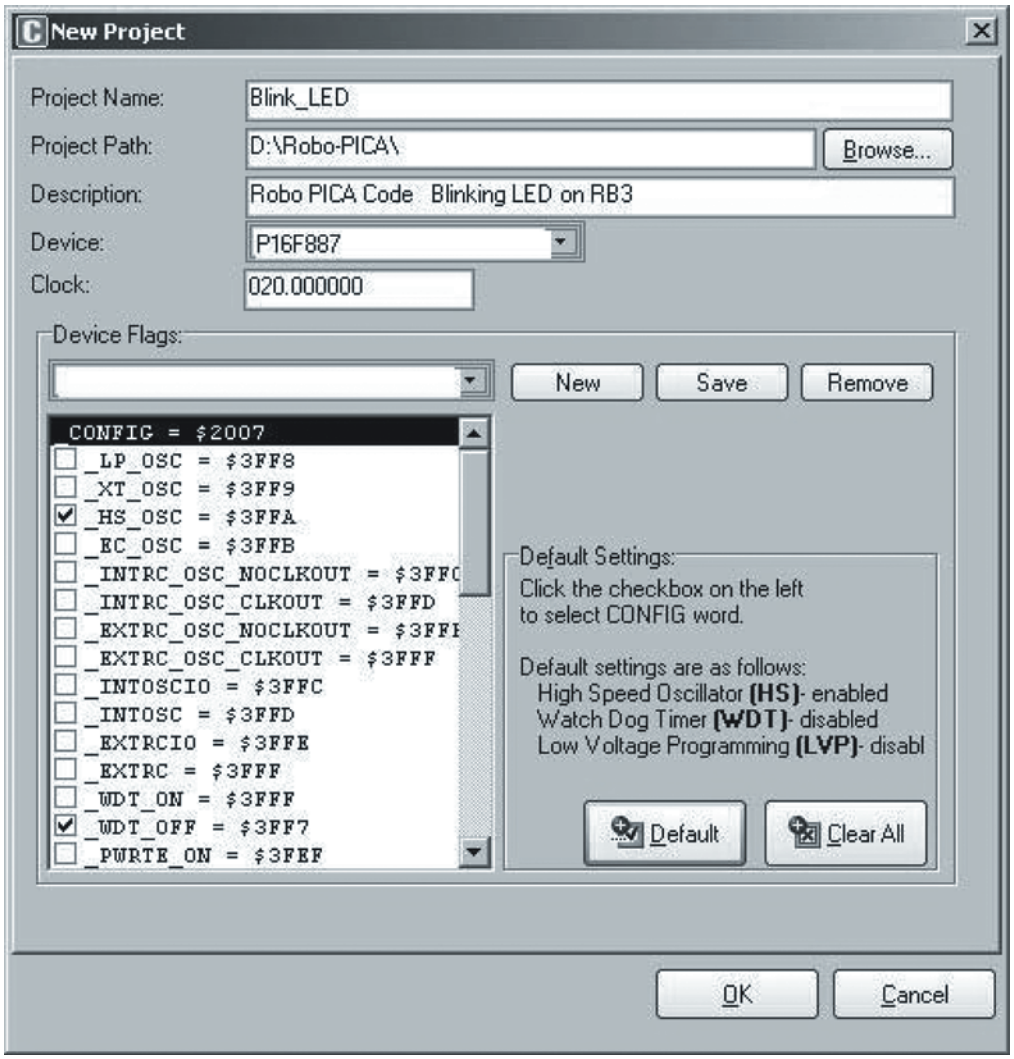


Рис. 1-1. Главное окно графической оболочки MikroC IDE и ее важные компоненты, которые необходимо знать.

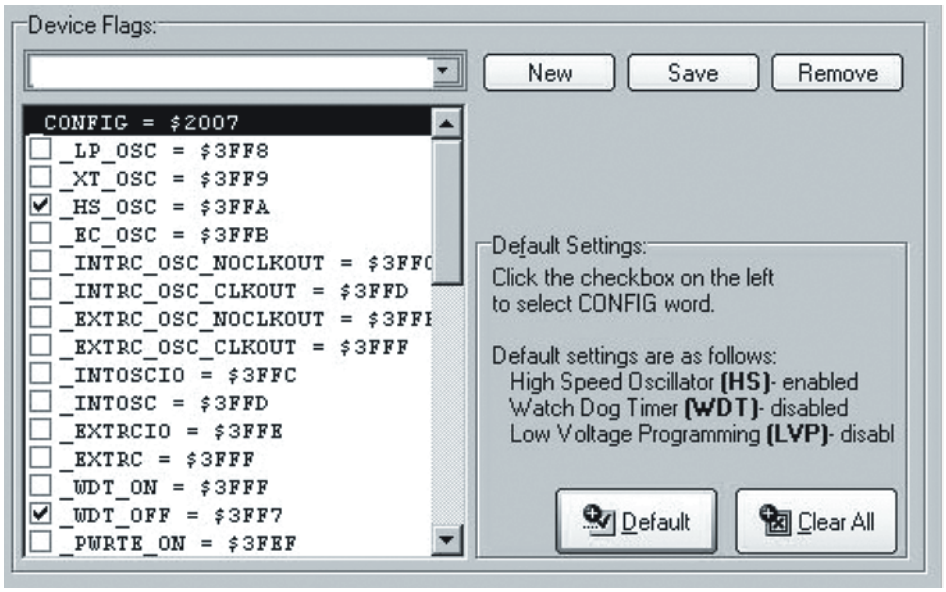
1.5.4 Создать новый файл проекта, зайдя в меню **Project (Проект)** и выбрав пункт **New Project...(Новый Проект...)**



1.5.5 В появившемся окне **New Project (Новый Проект)** необходимо определить важные параметры проекта, как это показано на следующем рисунке:



a. **Project Name (Имя Проекта)**: Введите имя проекта. MikroC IDE создаст каталог, в котором будут храниться все файлы проекта, включая исходные тексты на языке C. Например, файл проекта ***Blink_LED***.
b. **Project Path (Каталог Проекта)**: Выберите каталог, в котором будет создан Ваш проект. Нажмите кнопку **Browse (Обзор),** чтобы выбрать путь к каталогу. Например, ***D:\ROBO-PICA***
c. **Description (Описание)**: Введите информацию о Вашем файле проекта. Например ***"Robo PICA Code Blinking LED on RB3" ("Robo PICA мигающий светодиод на выводе RB3")***
d. **Device (Микросхема)**: Выберите микроконтроллер, который будете использовать в проекте. Для набора Robo-PICA и платы Управления RBX-877V2.0 необходимо выбрать **PIC16F887**
e. **Clock**: Выберите частоту тактового генератора для используемого микроконтроллера. Для набора Robo-PICA и платы Управления RBX-877V2.0 используется тактовый генератор частотой 20 МГц. Введите значение **020.000000.**
f. **Device Flags (Флаги микросхемы)**: Установка конфигурационных флагов используемого микроконтроллера. Разработчик может ограничиться установкой флагов по умолчанию, нажав кнопку Default  (По умолчанию). По умолчанию устанавливаются следующие 3 конфигурационных бита:



Разрешено использование Высокочастотного Генератора (HS_OSC) для частоты тактового генератора 10 МГц и выше.

Сторожевой Таймер Отключен (WDT_OFF)

Низковольтное Программирование Отключено (LVP_OFF)

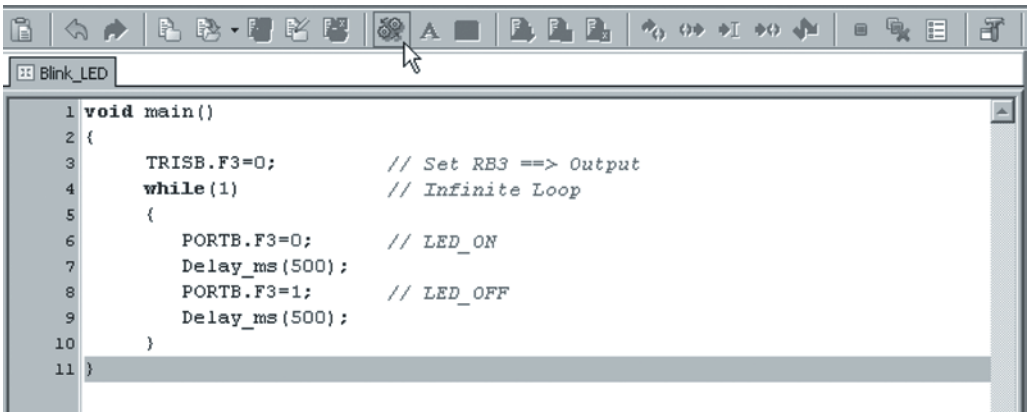
После завершения конфигурирования, нажмите кнопку **OK**. MikroC IDE закроет окно **New Project** и создаст файл ***Blink_LED.C*** с пустой областью редактирования для написания программы на языке C.

1.5.6 Введите текст программы на языке C, следуя распечатке 1-1.

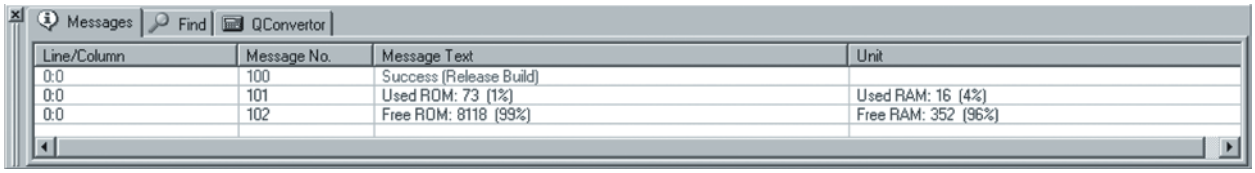
```
void main()
{
    TRISB.F3=0; // Использовать вывод RB3 как Выход
    while(1) // Бесконечный цикл
    {
        PORTB.F3=0; // Светодиод включен
        Delay_ms(500);
        PORTB.F3=1; // Светодиод выключен
        Delay_ms(500);
    }
}
```

Распечатка 1-1. Текст тестовой программы для мигающего светодиода.

1.5.7 Нажмите кнопку **Build Project (Собрать Проект)** или используйте комбинацию клавиш **Ctrl+F9**, чтобы скомпилировать файл проекта.

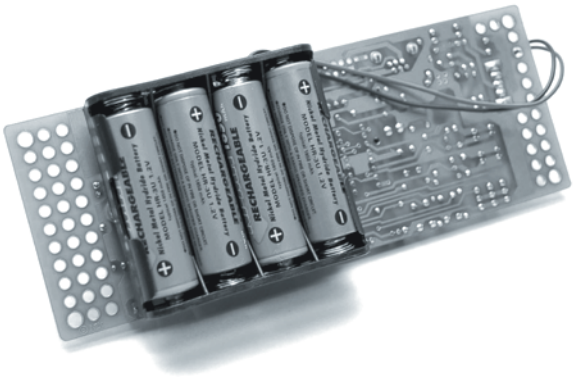


1.5.8 Просмотрите сообщения об ошибках в **Output window (Выходном окне)**. Если ошибки отсутствуют, то Вы увидите размер используемой этим файлом программной памяти микроконтроллера и сообщение об успешном завершении компиляции.

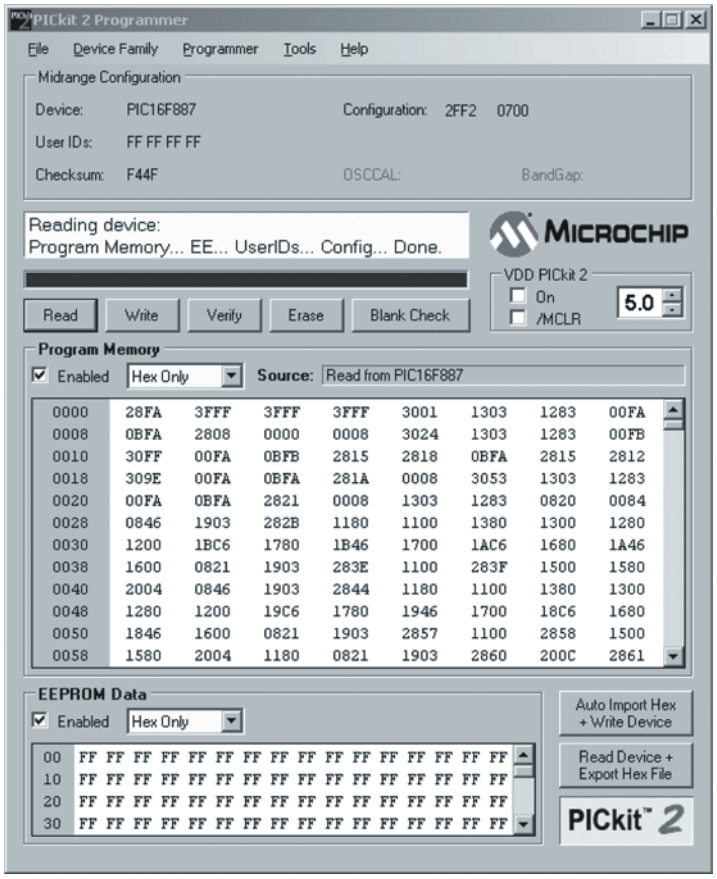


После этого, можно использовать полученный HEX-файл; ***Blink_LED.hex*** для загрузки в плату управления RBX-877V2.0 набора Robo-PICA.

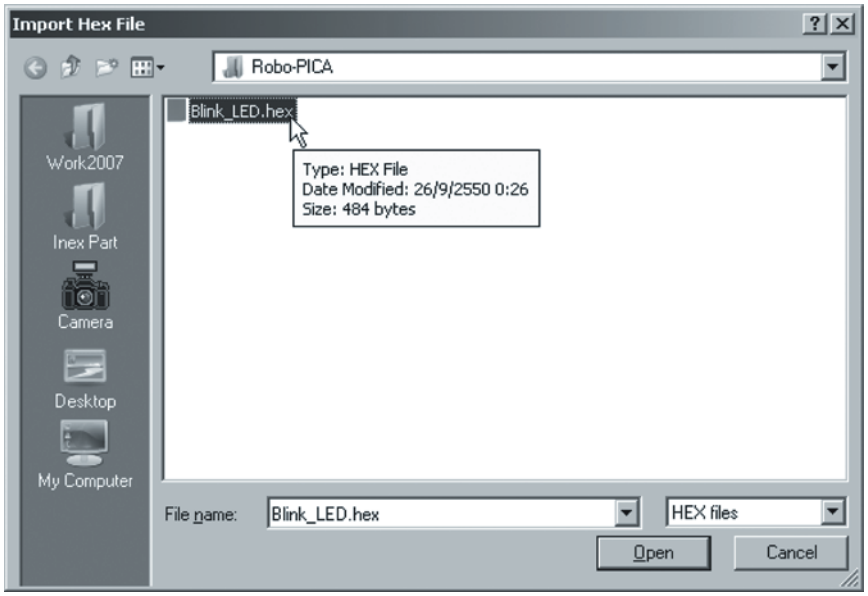
1.5.9 Вставьте 4 батарейки типа AA в батарейный отсек платы Управления RBX-877V2.0.



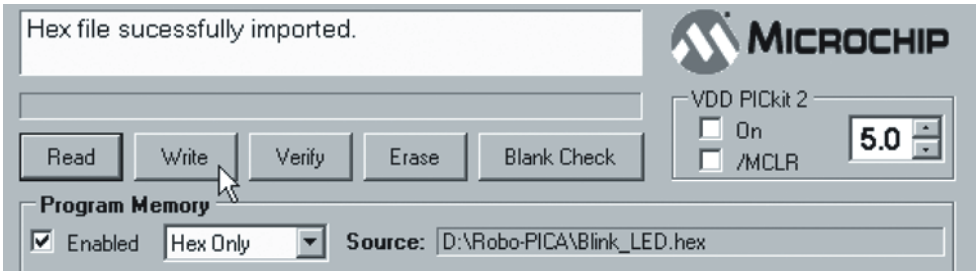
1.5.10 Присоедините USB-программатор к USB-порту компьютера.
1.5.11 Соедините ICD2-кабелем USB-программатор и плату Управления RBX-877V2.0.
1.5.12 Включите питание платы Управления RBX-877 V2.0.
1.5.13 Откройте Программное обеспечение PICkit2™.
1.5.14 Если все соединения выполнены правильно, Программное обеспечение PICkit2™ автоматически определит тип программируемого микроконтроллера и выдаст сообщение **PIC16F887 is found (Обнаружен микроконтроллер PIC16F887).**



1.5.15 Выберите HEX, который необходимо запрограммировать в микроконтроллер, войдя в меню **File → Import Hex**. При этом появится окно загрузки HEX-файла. Войдите в каталог ***D:\ROBO-PICA\Blink_LED*** и выберите файл ***Blink_LED.hex***



1.5.16 Нажмите кнопку **Write (Записать),** чтобы загрузить HEX-файл в плату Управления RBX-877 V2.0.



1.5.17 Пронаблюдайте результат на светодиоде, подключенном к выводу RB3 платы Управления RBX-877 V2.0. Светодиод, подключенный к выводу ***RB3 платы Управления RBX-877V2.0, непрерывно мигает!***

Глава 2

Плата Управления Роботом RBX-877V2.0

Набор для изготовления робота Robo-PICA управляется RBX-877 V2.0 (Плата для Робототехнических Экспериментов на микроконтроллере PIC16F887). Главным микроконтроллером в наборе является микросхема PIC16F887. На рисунке 2-1 показана диаграмма функционирования платы RBX-877. В этой главе будет детально описана работа платы RBX-877 и некоторые примеры экспериментов с ней. Для успешной сборки робота необходимо прочитать описание всех экспериментов и проверить их на практике, чтобы перейти, непосредственно, к сборке робота, описанной в следующей главе.

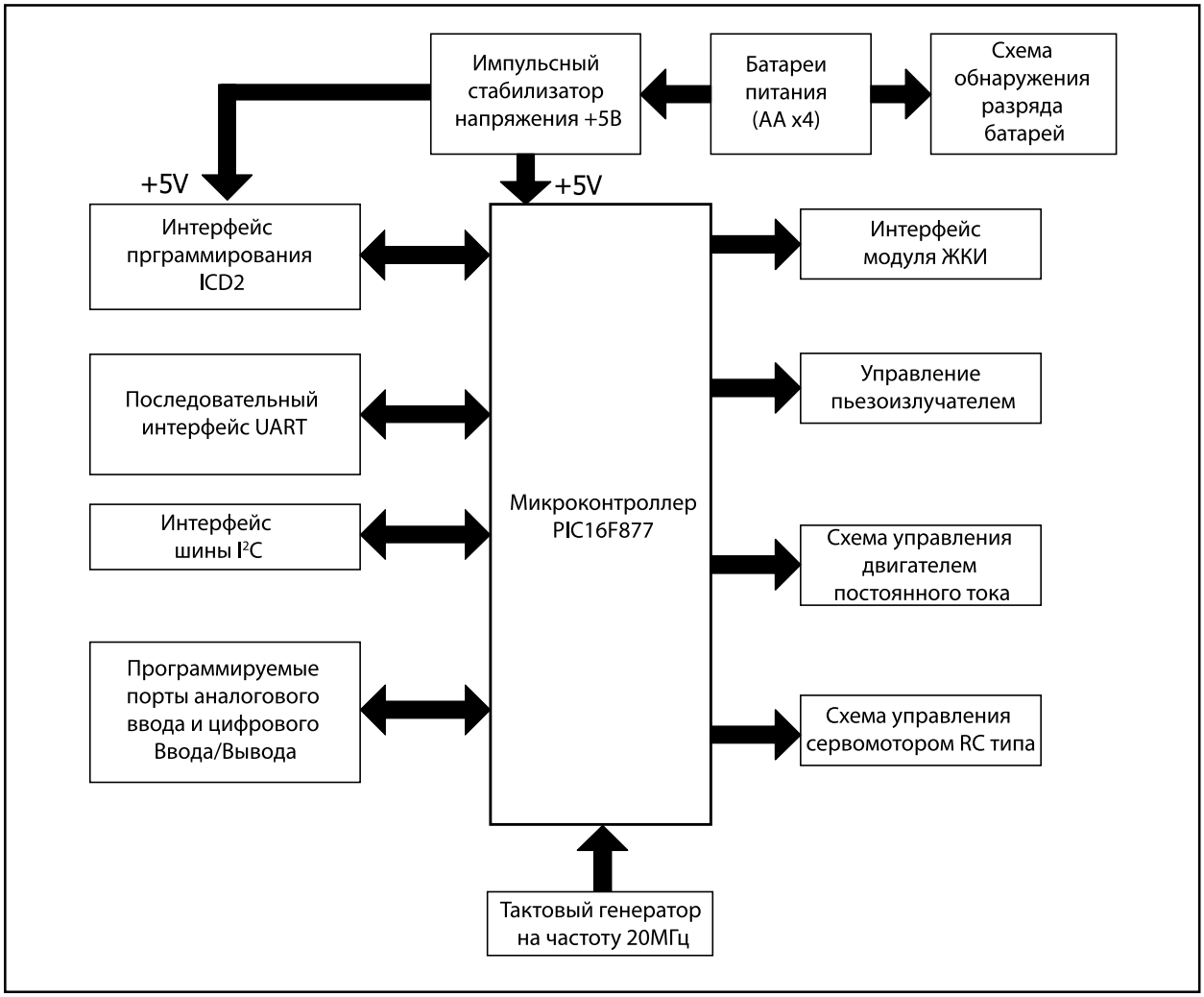


Рис 2-1. Блок-диаграмма платы управления роботом RBX-877 V2.0.

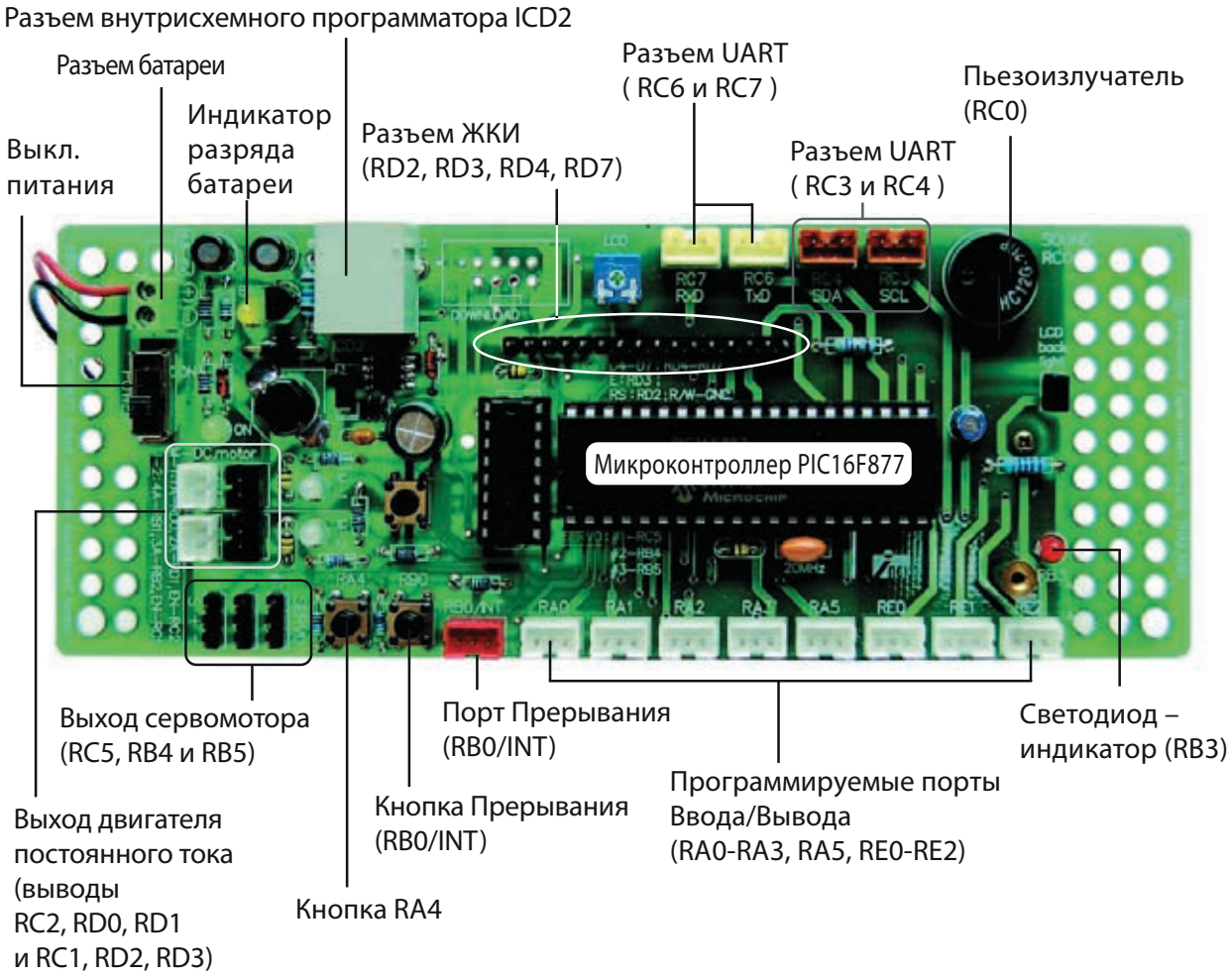


Рис 2-1. Расположение элементов на плате управления роботом RBX-877 V2.0.

2.1 Технические данные платы RBX-877 V2.0

- Управляется микроконтроллером PIC16F887 с объемом памяти микропрограмм 8 кСлов. Работает на тактовой частоте 20 МГц
- Микропрограммы загружаются через разъем ICD2.
- ЖКИ-индикатор16x2 со светодиодной подсветкой и перемычкой включения/выключения подсветки
- Пьезоизлучатель
- Светодиодный индикатор включения
- Управление двумя двигателями постоянного тока, с напряжением питания 4,5...6 В и тремя сервомоторами RC типа (в диапазоне от 4,8 до 6 В)
- 9 программируемых портов, поддерживающих как аналоговый ввод, так и цифровой ввод/вывод
- Порт шины I^2C
- Порт UART для организации обмена с последовательными устройствами, такими как: приемопередатчики шины RS-232, модули XBee и Bluetooth.
- Питание осуществляется от 4-х пальчиковых батареек размера AA (рекомендуется использовать перезаряжаемые аккумуляторы)
- Размер платы 60x158 мм (2,375 x 6,25 дюйма)

2.2 Описание принципиальной схемы платы RBX-877 V2.0

2.2.1 Блок микроконтроллера

Сердцем этой платы является микроконтроллер PIC16F887. Чтобы получить тактовую частоту 20 МГц для PIC16F887, используется керамический резонатор CR1 на частоту 20 МГц.

2.2.2 Источник питания

Плата RBX-877 V2.0 содержит повышающий импульсный источник питания, чтобы обеспечить регулируемое напряжение +5 В для PIC16F887. Несмотря на то, что уровень напряжения батарей будет уменьшаться при управлении двигателями, схема импульсного источника питания сможет обеспечивать постоянное напряжение +5 В для питания микроконтроллера до тех пор, пока уровень напряжения батарей не упадет ниже 1,5 В.

Выключатель S1 позволяет включать/выключать напряжение питания, идущее от батарей к плате RBX-877 V2.0. Элементы R3, D1 и ZD1 используются для ограничения входного напряжения для IC2 на уровне 5,1 В.

IC1 – это микросхема управления импульсным источником питания NCP1450-5.0. Она может работать при входном напряжении от 1,5 до 4,2 В, выдавая на выходе стабильное напряжение +5 В для питания микропроцессора. Стабилитрон ZD2 используется для ограничения выходного напряжения NCP1450-5.0 на уровне +5 В.

2.2.3 Блок Внутрисхемного Программирования

Для программирования платы RBx-877 V2.0 требуется внутрисхемный программатор, подключаемый к разъемам ICD2 или ISP. USB-программатор, который входит в комплект набора Robo-PICA, присоединяется к плате управления RBX-877 V2.0 посредством ICD2-разъема. Он использует напряжение питания непосредственно от USB-порта компьютера.

Сигнал программирования будет поступать на выводы RB6 и RB7 микроконтроллера PIC16F887. Высокое напряжение для программирования подается на вывод MCLR. Все стадии операции программирования можно пронаблюдать в окне Программного обеспечения PICkit2 на мониторе компьютера. После завершения операции программирования, этот контроллер может внезапно заработать без подачи каких-либо внешних команд.

2.2.4 Блок Отображения информации

Символьный Индикатор: Плата RBX-877 V2.0 имеет разъем для подключения модуля ЖКИ. Она может работать с ЖКИ-индикатором размером 16 символов в 2 строках, который подключается к выводам RD4...RD7 (сигналы управления ЖКИ) и выводам D4...D7 (шина данных ЖКИ) микроконтроллера PIC16F877, вывод RD3 используется как сигнал E и вывод RD2 как сигнал RS для выбора режима обмена данными с ЖКИ. Потенциометр VR1 используется для изменения яркости экрана ЖКИ. В случае использования ЖКИ с фоновой подсветкой, имеется перемычка для управления светодиодом фоновой подсветки ЖКИ.

Светодиодный монитор: Плата RBX-877 V2.0 имеет светодиод общего назначения. Он присоединен к выводу RB3 микроконтроллера PIC16F887 через токоограничивающий резистор.

Звуковой выход: Плата RBX-877 V2.0 имеет схему управления звуком. К выводу RC0 через разделительный конденсатор емкостью 10 мкФ подключен пьезоизлучатель. Этот блок может генерировать сигналы звуковой частоты. Однако пьезоизлучатель имеет диапазон резонансной частоты 1...3кГц, что накладывает ограничения на возможность воспроизведения звукового сигнала.

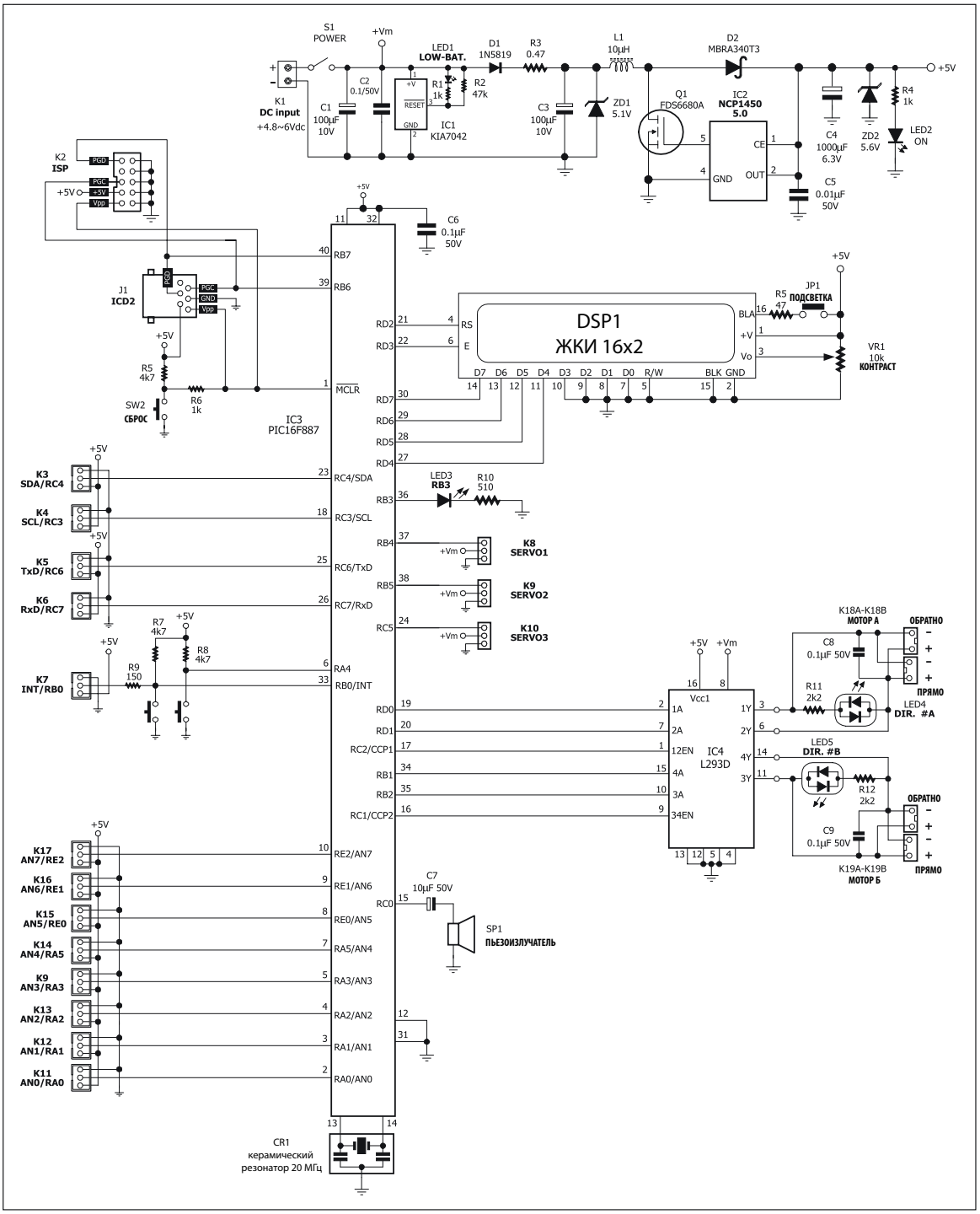


Рис 2-3. Принципиальная схема платы управления роботом RBX-877 V2.0.

2.2.5 Программируемый порт

Плата RBX-877 V2.0 имеет 9-программируемых многофункциональных портов. Они включают в себя выводы RA0-RA3, RA5, RE0-RE2 и RB0. Все порты могут быть запрограммированы для выполнения любой из следующих 3 функций:

(1) Аналоговый вход – чтобы отправить аналоговый сигнал на схему аналого-цифрового преобразователя внутри микроконтроллера. Входное напряжение должно находиться в диапазоне 0...5 В. Разрешающая способность преобразователя составляет 10 разрядов.

(2) Цифровой вход – чтобы принять цифровой сигнал от цифрового устройства или переключателя (кнопки).

(3) Цифровой выход – чтобы передать цифровой логический сигнал ("0" или"1") внешнему устройству.

По умолчанию все порты сконфигурированы как порты аналогового ввода.

На плате RBX-877 V2.0 все порты выведены на 3-контактные JST-разъемы. На каждом разъеме, кроме сигнального вывода присутствуют выводы напряжения питания +5 В и общего провода GND.

2.2.6 Порт UART для последовательного проводного/беспроводного обмена данными

Разработчик может организовать последовательный обмен данными между платой RBX-877 V2.0 и компьютерным последовательным портом RS-232, и многими беспроводными последовательными устройствами, такими как модули XBEE и Bluetooth. Микроконтроллер PIC16F887 использует выводы RC6 и RC7 как выводы модуля UART. Последовательный сигнал от микроконтроллера PIC16F887 выведен на 2 свободных JST-разъема для поддержки обмена с любым последовательным устройством.

2.2.7 Блок управления двигателем постоянного тока

Плата RBX-877 V2.0 использует IC4, типа L293D, ИС-полумостовой схемы управления двигателями для управления 2 каналами двигателей постоянного тока. Рабочее напряжение двигателей может находиться в пределах 4,5-6 В, рабочий ток 100-200 мА, в пределе до 400 мА.

Частота вращения двигателя A управляется выводами RD0 и RD1 и выводом разрешения вращения RC2. Частота вращения двигателя B управляется выводами RB1 и RB2 и выводом разрешения вращения RC1. Светодиоды LED4 и LED5 – двухцветные. Они используются для индикации выходного состояния двигателей.

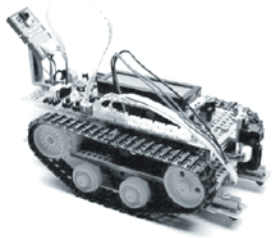
Напряжение питания L293D включает напряжение питания ядра микросхемы +5 В и напряжения питания выходных каскадов для управления двигателями (+Vm). Вывод +Vm присоединен непосредственно к батареям для лучшего использования источников питания.

2.2.8 Блок управления RC-сервомоторами

Плата Управления RBX-877 V2.0 имеет выводы 3 портов для RC-сервомоторов. Они включают в себя выводы RB4, RB5 и RC5. RC-сервомоторы запитываются от системных батарей. Этот блок не поддерживает RC-сервомоторы с высоким потребляемым током и высокой потребляемой мощностью. Типичным является управление сервомоторами RC с напряжением питания от 4,8 до 6 В и потребляемым током примерно 100-200 мА.

2.2.9 Разъем шины I^2C

Одним из путей расширения функциональных возможностей платы RBX-877 V2.0 является использование разъема шины I^2C. Для работы со многими внешними устройствами необходимо наличие обмена по протоколу шины I^2C. К таким устройствам относятся часы реального времени, микросхемы памяти, аналого-цифровые и цифро-аналоговые преобразователи, микросхемы расширителей портов ит.п. Сигналы RC3/SCL и RC4/SDA-микроконтроллера PIC16F887 выведены на разъем шины I^2C, включающий контакты напряжения питания +5 В и общего провода GND. К этому порту нельзя подключать подтягивающие резисторы. Пользователь должен установить их на внешнем устройстве.



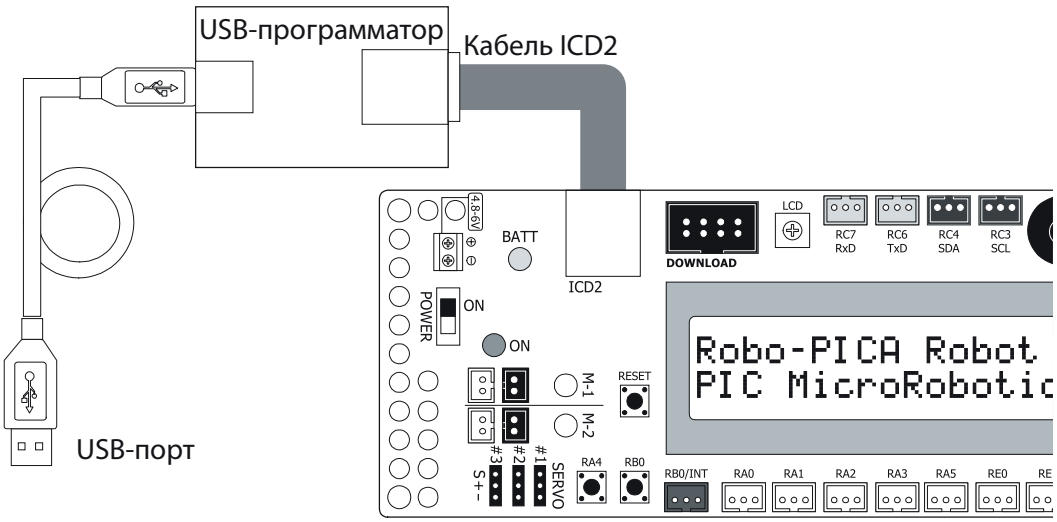
Задание 1

Написание программы для тестирования платы управления RBX-877 V2.0

Последовательность выполнения

Для выполнения всех заданий по программным разработкам для робототехнического набора Robo-PICA рекомендуется следующая последовательность выполнения шагов:

1. Создать проект на языке C с использованием программы mikroC IDE.
2. Скомпилировать файл проекта.
3. При наличии ошибок, редактировать программу на языке C, чтобы исправить найденные ошибки до тех пор, пока все ошибки не будут исправлены.
4. После успешной компиляции должен быть создан выходной HEX-файл.
5. Присоединить USB-программатор к USB-порту и подключить ICD2-кабель между USB-программатором и ICD2-разъемом платы управления RBX-877 V2.0.



6. Открыть программное обеспечение PICkit2™ и проверить наличие обмена.
7. Загрузить HEX-файл в PIC16F887 на плате управления RBX-877 V2.0 набора Robot-PICA.
8. Запустить программу на исполнение и проверить правильность выполнения операций. При наличии ошибок, вернуться к редактированию программы на языке C, скомпилировать HEX-файл и загрузить еще раз. Выполнять указанные шаги до тех пор, пока операции не будут выполнены полностью.

Задание 1-1 Тестирование светодиода

Обратитесь к рисунку А1-1. К выводу RB3-микроконтроллера PIC16F887 присоединен светодиод через токоограничивающий резистор сопротивлением 510 Ом. Для зажигания светодиода необходимо на этом выводе установить уровень логической "1" и установить уровень логического "0", чтобы светодиод погас.

А1.1.1 Напишите программу, следуя распечатке А1-1, затем скомпилируйте ее и загрузите в плату управления Роботом RBX-877 V2.0. Пронаблюдайте результат работы.

Светодиод, подключенный к выводу RB3, зажегся.

А1.1.2 Напишите программу, следуя Распечатке А1-2, затем скомпилируйте ее и загрузите в плату Управления Роботом RBX-877 V2.0. Пронаблюдайте результат работы.

Светодиод, подключенный к выводу RB3, замигал с интервалом 0,5 сек.

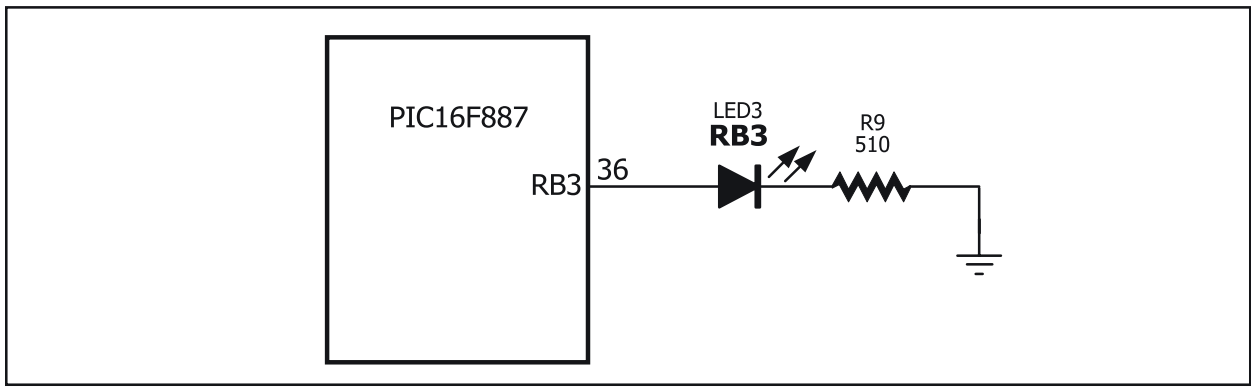


Рис. А1-1: Подключение светодиода к плате управления Роботом RBX-877 V2.0

```
void main()
{
     TRISB.F3=0; // Вывод RB3 ==> Выход
     PORTB.F3=1; // Включить RB3
}
```

Распечатка А1-1: Программа на языке С для включения светодиода, подключенного к выводу RB3 платы Управления Роботом RBX-877 V2.0

```
void main()
{
    TRISB.F3=0; // Вывод RB3 ==> Выход
    while(1)
    {
        PORTB.F3=1; // Включить RB3
        Delay_ms(500);
        PORTB.F3=0; // Выключить RB3
        Delay_ms(500);
    }
}
```

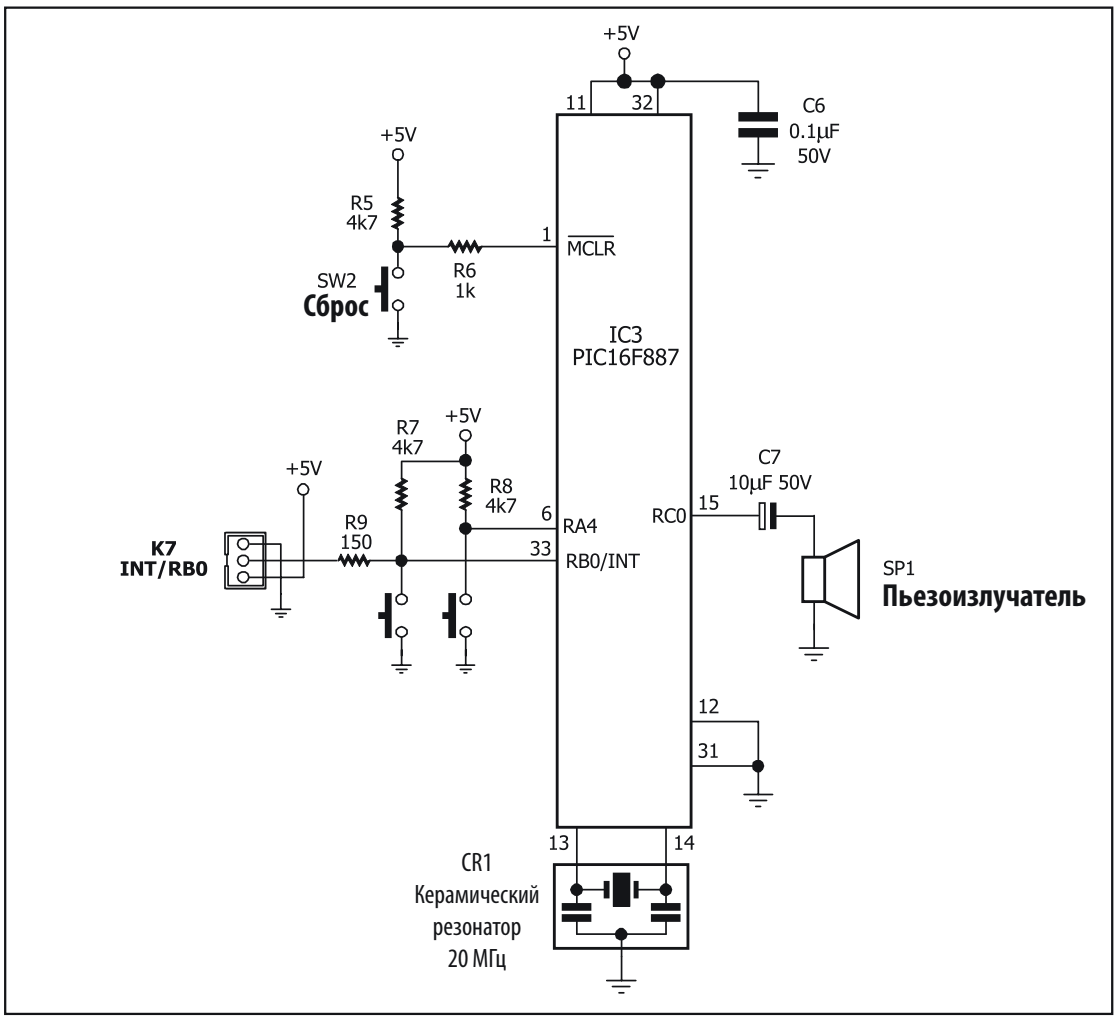
Распечатка А1-2: Программа на языке С мигания светодиода, подключенного к выводу RB3 платы Управления Роботом RBX-877 V2.0

Задание 1-2 Чтение цифровых данных о состоянии переключателя и управление звуком

На рисунке A1-2 показана принципиальная схема подключения кнопки к плате Управления RBX-877 V2.0. В этой лабораторной работе используется кнопка, подключенная к порту RA4. При отсутствии нажатия на кнопку, на вход подается сигнал, соответствующий уровню логической "1", от подтягивающего резистора сопротивлением 10 кОм, подключенного к источнику питания +5 В. При нажатии на кнопку, вход будет соединен с общим проводом схемы. В этом случае на входе микроконтроллера будет присутствовать сигнал, соответствующий логическому уровню "0". Микроконтроллер PIC16F887 будет управлять подачей звукового сигнала в соответствии с состоянием кнопки, подключенной к выводу RA4.

Программирование чтения состояния кнопки

Простейшим способом определить состояние кнопки будет организация в программе на языке C в среде компилятора mikroC бесконечного цикла и проверка с использованием оператора IF. Если кнопка нажата, программа будет переходить к выполнению некоторого действия. При написании программы, в первую очередь необходимо выбрать порт, который будет обрабатывать состояние кнопки.



A1-2: Схема включения вывода RA4 платы контроллера RBX-877 V2.0

Проверка

A1.2.1 Введите текст, показанный на распечатке A1-3. Скомпилируйте и загрузите код в плату RBX-877.

A1.2.2 Нажмите кнопку, подключенную к порту RA4, и пронаблюдайте работу пьезоизлучателя на плате Управления Роботом RBX-877 V2.0.

При нажатии на кнопку Вы должны услышать звук из пьезоизлучателя.

```
void main()
{
    Sound_Init(&PORTC, 0); // Инициализация Звукового сигнала
    while(1)
    {
        if (!PORTA.F4) // Тестирование состояния кнопки RA4
        sound_play(250,50); // Выдача звукового сигнала 2кГц на вывод RC0
    }
}
```

Распечатка A1-3: Программа на языке C для чтения цифрового значения со входа RA4, к которому подключена кнопка для управления генерацией звука пьезоизлучателем, подключенным к выводу RC0. Данную операцию можно использовать для индикации открывания двери.

Задание 1-3. Отображение сообщения на ЖКИ-индикаторе

Плата Управления Роботом RBX-877 V2.0 имеет разъем для подключения модуля ЖКИ. Схема подключения модуля ЖКИ показана на рисунке A1-3. Пользователю необходимо использовать эту информацию, чтобы в программе на языке C для компилятора mikroC назначить выводы порта для использования этого подключения.

Для подключения ЖКИ необходимо 6 выводов порта: вывод RD2 для сигнала RS ЖКИ-модуля, вывод RD3 для сигнала "E" ЖКИ модуля и выводы RD4...RD7 для сигналов данных D4...D7 при использовании четырехпроводного режима передачи данных. Вывод R/W подключается к общему проводу, разрешая только операцию записи данных в ЖКИ. Такое соединение помогает разработчику использовать более простой код на языке C для управления модулем ЖКИ. При этом можно использовать встроенную в компилятор mikroC функцию управления ЖКИ: **cd_Init(&PORTD)**.

Проверка

A1.3.1 Введите текст, показанный на распечатке A1-4. Скомпилируйте и загрузите код в плату RBX-877.

A1.3.2 Пронаблюдайте работу ЖКИ-модуля.

На ЖКИ-модуле должно появиться сообщение **Innovative** *в верхней строке и сообщение* **Experiment** *в нижней строке. При необходимости использования фоновой подсветки ЖКИ, установите перемычку в положение "LCD backlight".*

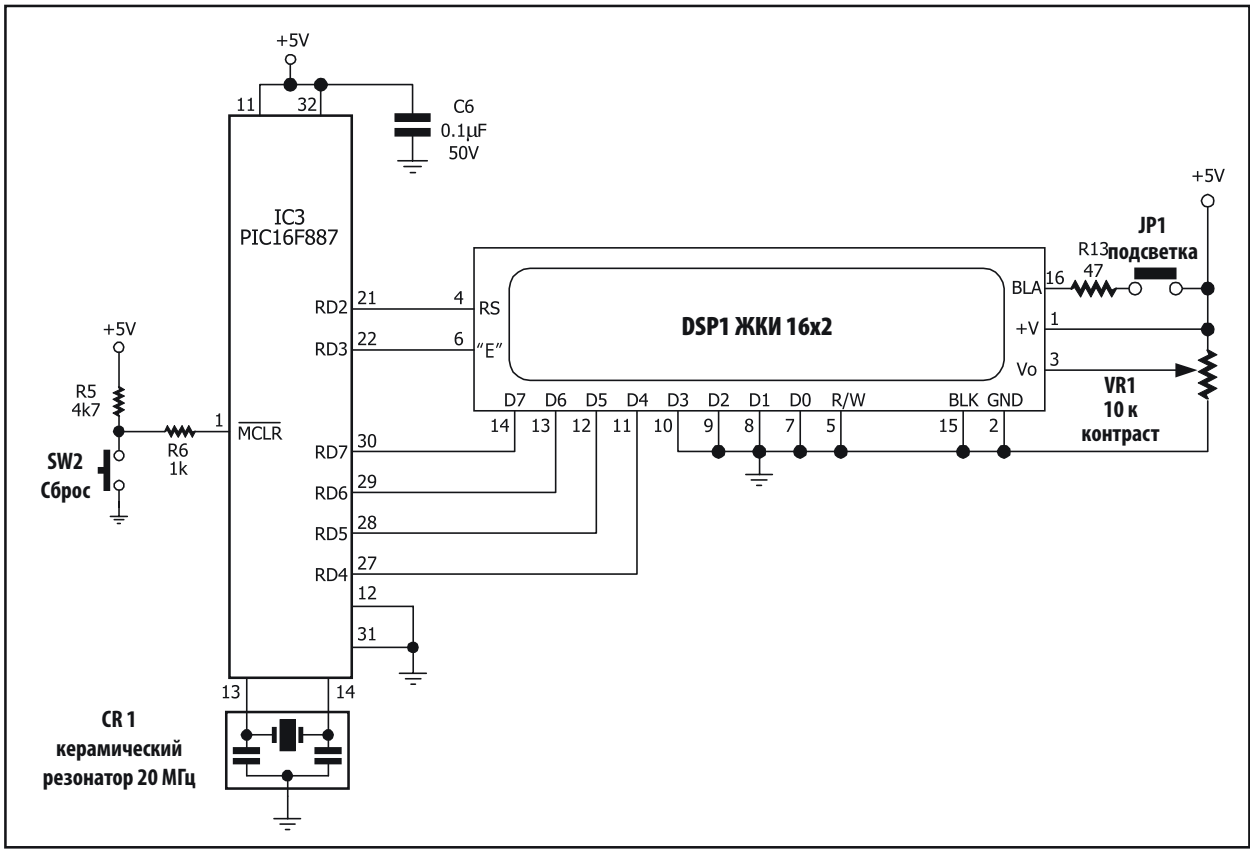


Рис. А1-3. Принципиальная схема ЖКИ-интерфейса платы RBX-877.

```
char *text1 = "Innovative";
char *text2 = "Experiment";

void main()
{
  Lcd_Init(&PORTD);
  Lcd_Cmd(LCD_CURSOR_OFF);
  while(1)
  {
    Lcd_Out(1,1,text1);
    Lcd_Out(2,1,text2);
    Delay_ms(5000);
    Lcd_Cmd(LCD_CLEAR);
    Delay_ms(500);
  }
}
```

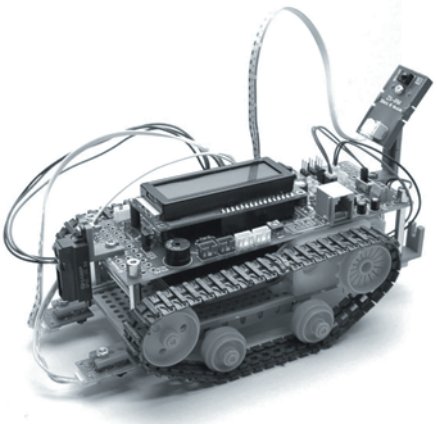
Распечатка А1-4. Программа на языке С для отображения сообщения на ЖКИ-модуле платы управления Роботом RBX-877 V2.0.

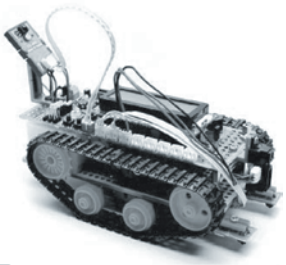
Глава 3

Сборка робота из набора Robo-PICA

Эта глава описывает как из набора Robo-PICA собрать действующего робота. Робот, собранный из набора Robo-PICA, обладает следующими характеристиками:

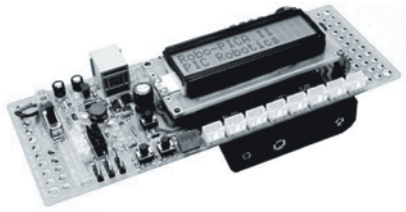
- Приводится в движение двумя коробками передач с двигателями постоянного тока и гусеницами
- Управляется микроконтроллером PIC16F887
- Имеет 8 к слов программной памяти
- Допускает более 10,000 циклов перепрограммирования flash-памяти микропрограмм
- Поддерживает множество типов датчиков и детекторов, таких как:
    - **ZX-01** Плата концевых выключателей для обнаружения препятствий,
    - **ZX-03** Инфракрасная отражательная оптопара для движения вдоль линий и границ областей,
    - **ZX-IRM** Модуль инфракрасного приемника для удаленного управления,
    - **GP2D120** Инфракрасный измеритель расстояний,
    - **SRF05** Ультразвуковой датчик,
    - **CMPS03** Цифровой компас,
    - **Memsic2125** Датчик ускорения и многие другие...
- Для индикации режимов работы робота используется символьный ЖКИ-модуль 16x2 отдельный светодиод состояния





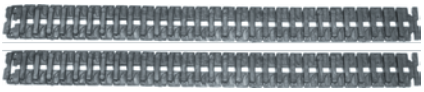
Задание 2

Сборка Robo-PICA



Плата управления RBX-877 на микроконтроллере PIC16F877

Металлическая ось (3шт.)



Фрагмент гусеницы из 30 звеньев (2 шт.)

Фрагмент гусеницы из 10 звеньев (4 шт.)

Винт M3x10 (15шт.)

Модуль двигателя постоянного тока (2 шт.)

Короткая уголковая планка (2 шт.)

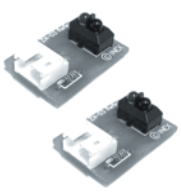
Заглушка (6 шт.)

Большое направляющие колесо (2шт.)

Винт с декоративной головкой (3 шт.)

Шуруп 2 мм (2 шт.)

Винт M3x15 (1 шт.)



Инфракрасные отражательные оптопары (2 шт)

Универсальная монтажная плата

Ведущее зубчатое колесо (2 шт.)

Шестигранная стойка длиной 30 мм (3 шт.)

Шайба 3 мм (2 шт.)

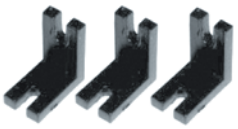
Плоская пластина с 3-мя отверстиями (2 шт.)

Гайка M3 (11шт.)

Модуль инфракрасного измерителя расстояний GP2D120

Длинная уголковая планка (2 шт.)

Среднее направляющие колесо (2 шт.)



Прямоугольная пластина (3 шт.)

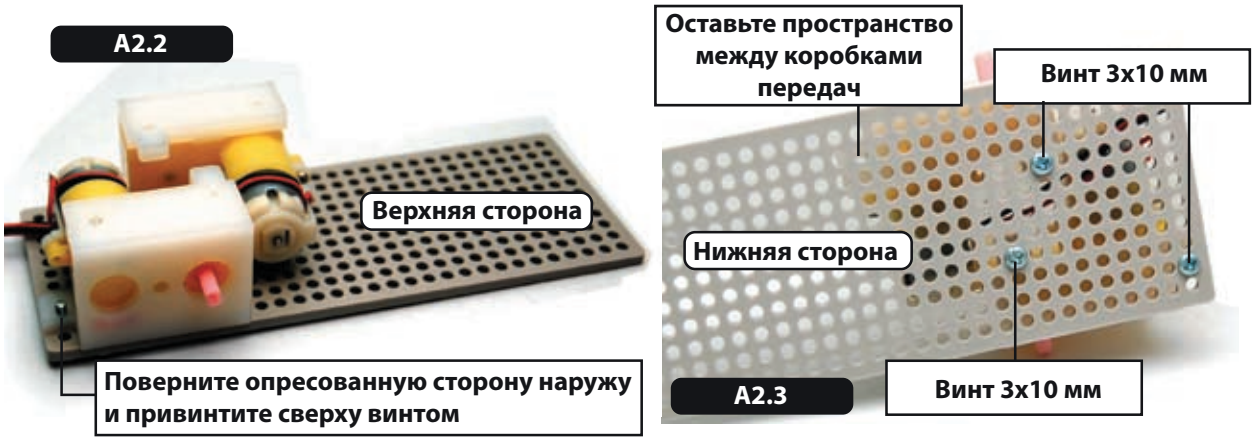
Тупоугольная пластина (3 шт.)

Плоская пластина (3 шт.)

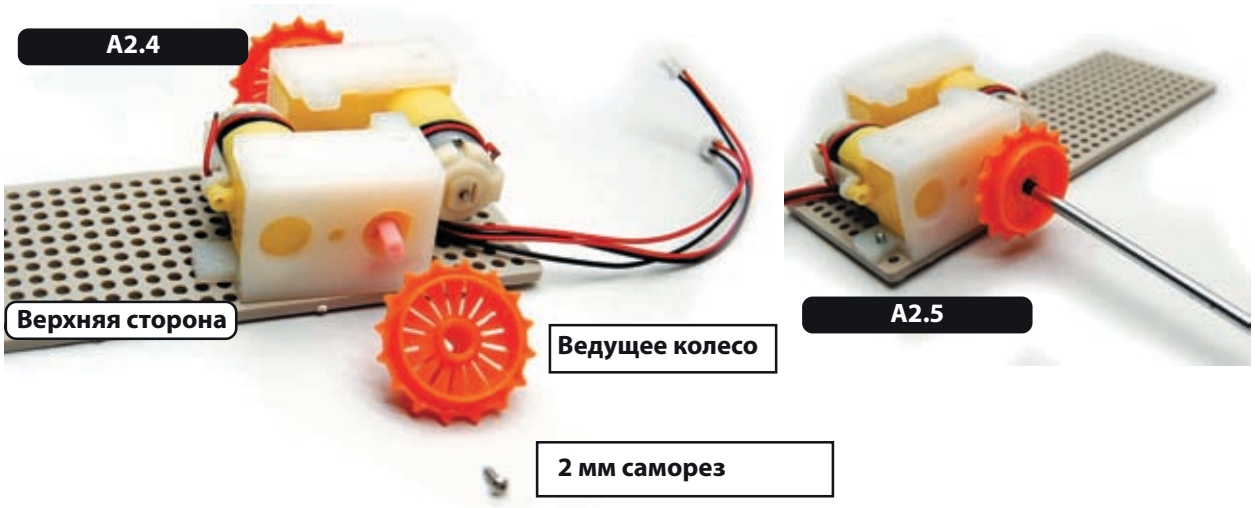
Модуль инфракрасного приемника на 38 кГц

A2-1: Комплектация набора Robo-PICA.

A2.1 Прикрепите 2 коробки передач с двигателями постоянного тока к основанию. Поверните опресованную сторону правой коробки передач наружу, как показано на рисунке A2-2. Привинтите коробку передач винтами 3x10 мм с верхней стороны к основанию так, чтобы осталось свободное пространство между левой коробкой передач. Не затягивайте винты слишком сильно.

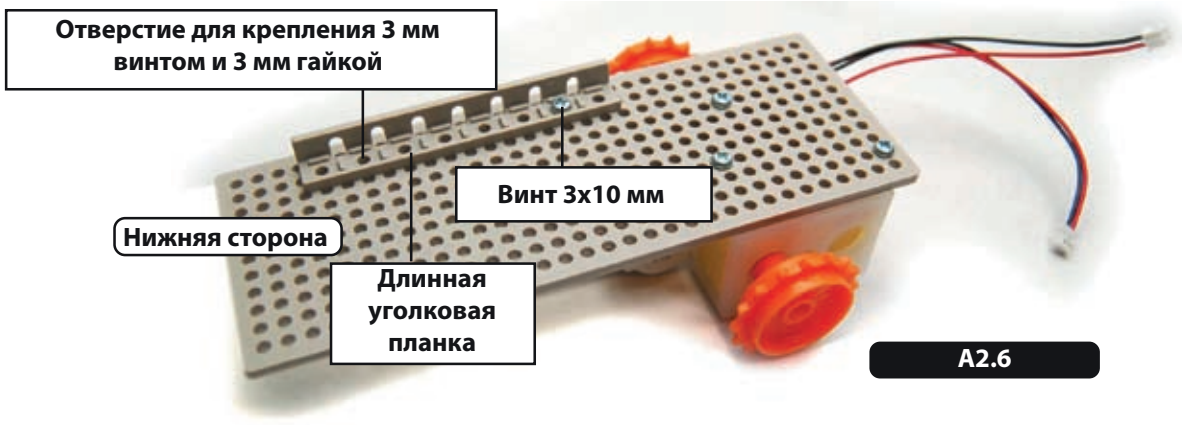


A2.2 Оденьте ведущее ходовое колесо на ось коробки передач и зафиксируйте его 2 мм саморезом. Проделайте эту операцию для обеих коробок передач.

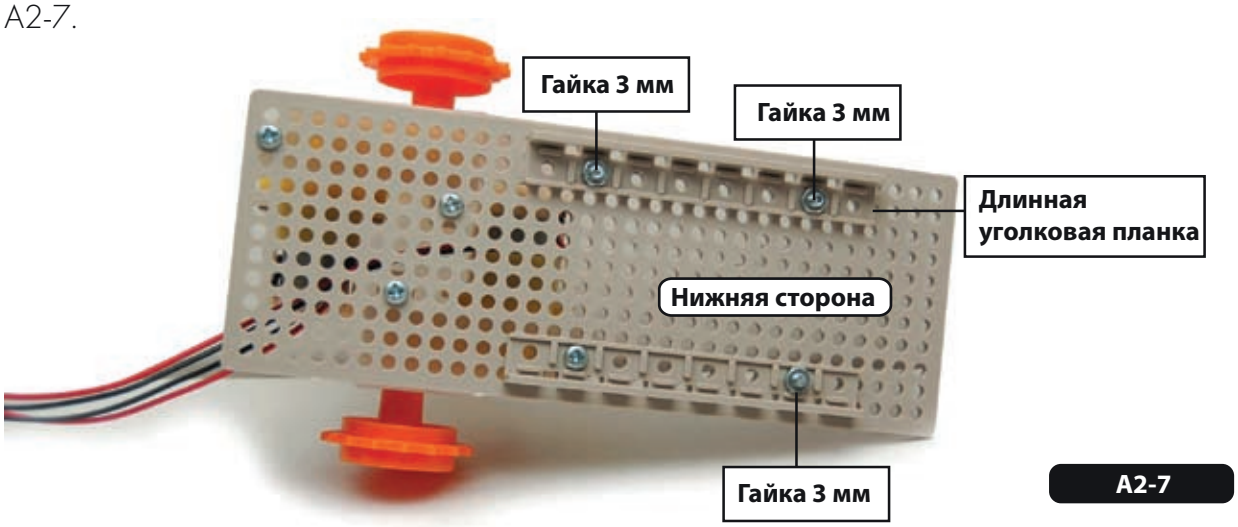


A2.3 Поверните основание нижней стороной вверх. Прижмите длинную уголковую планку к основанию в определенном положении, как показано на рисунке A2-6. Привинтите винтами 3x10 мм, предотвращая смещение планки относительно отверстий в панели.

A2.1 Завинтите еще один винт 3x10 мм и гайку 3 мм в другое отверстие длинной уголковой планки, как показано на рисунке A2-6.

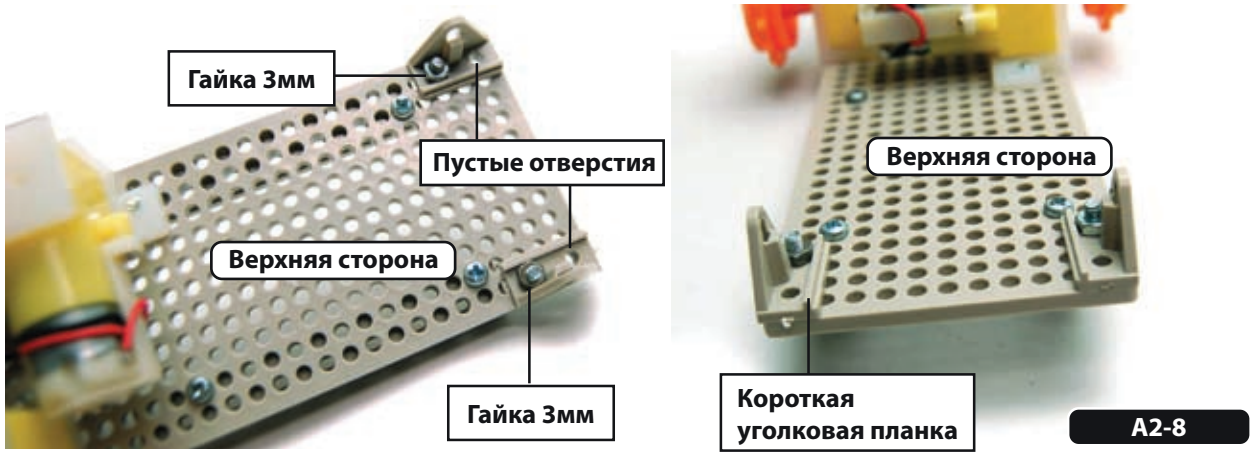


А2.4 Привинтите вторую длинную уголковую планку к основанию, вставив с верхней стороны винты 3x10 мм и затянув гайками 3 мм с нижней стороны, как показано на рисунке А2-7.



А2-7

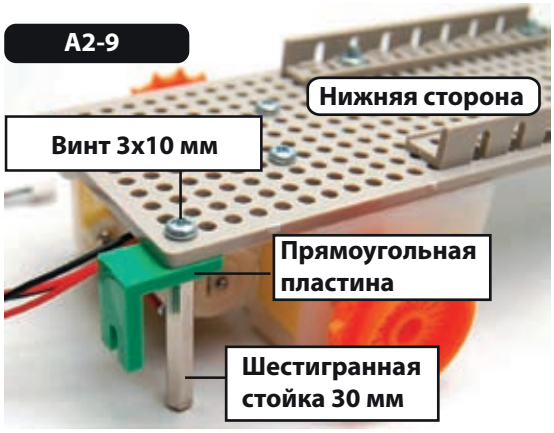
А2.5 Поверните основание в обратную сторону. Присоедините 2 коротких уголковых планки к задней стороне основания робота, как показано на рисунке А2-8, вставив винт 3x10 мм с нижней стороны основания через отверстие в планке и закрепив его 3 мм гайкой. Завинтите винт во внутреннем отверстии планки. Оставьте внешние отверстия пустыми.

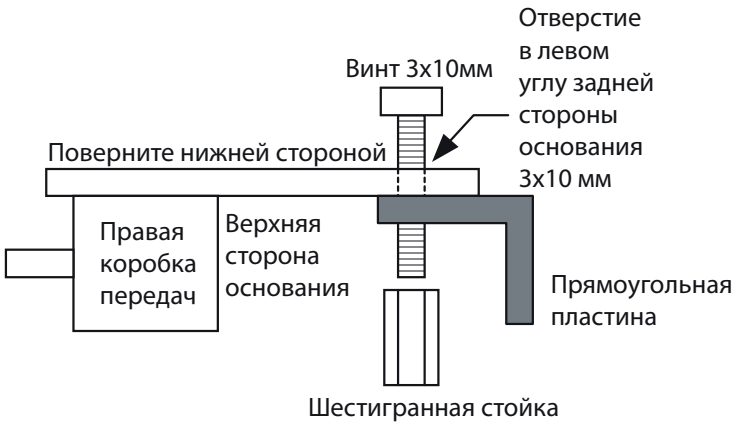


А2-8

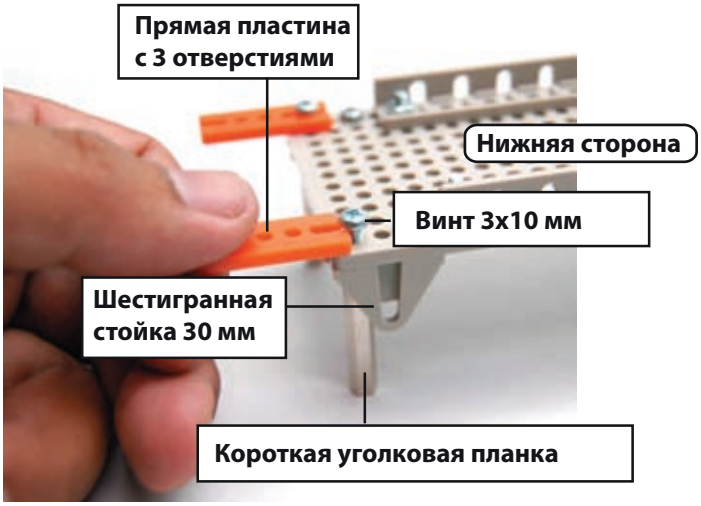
А2.6 Привинтите шестигранную стойку с верхней стороны основания, повернув верхнюю сторону вниз и пропустив винт 3x10 мм через левое угловое отверстие и прямоугольную пластину.

А2-9



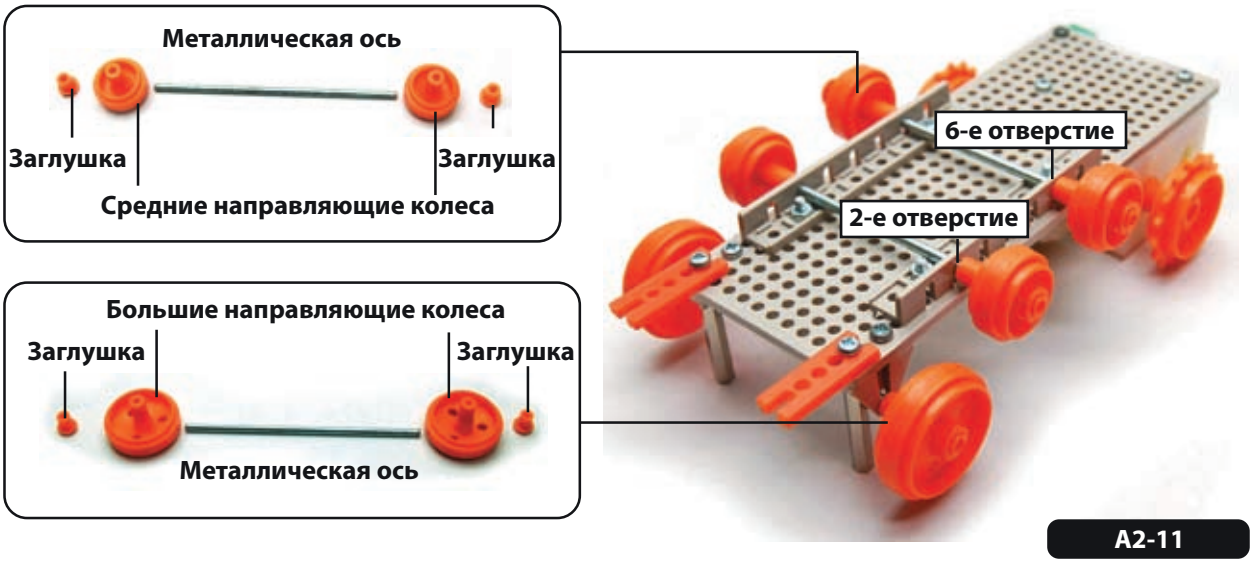


A2.7 Спереди присоедините 2 шестигранных стойки. Пропустите винт 3x10 мм через прямую пластину с 3 отверстиями и второе отверстие короткой уголковой планки, установленной на шаге A2.5, и зафиксируйте 30-миллиметровой шестигранной стойкой.



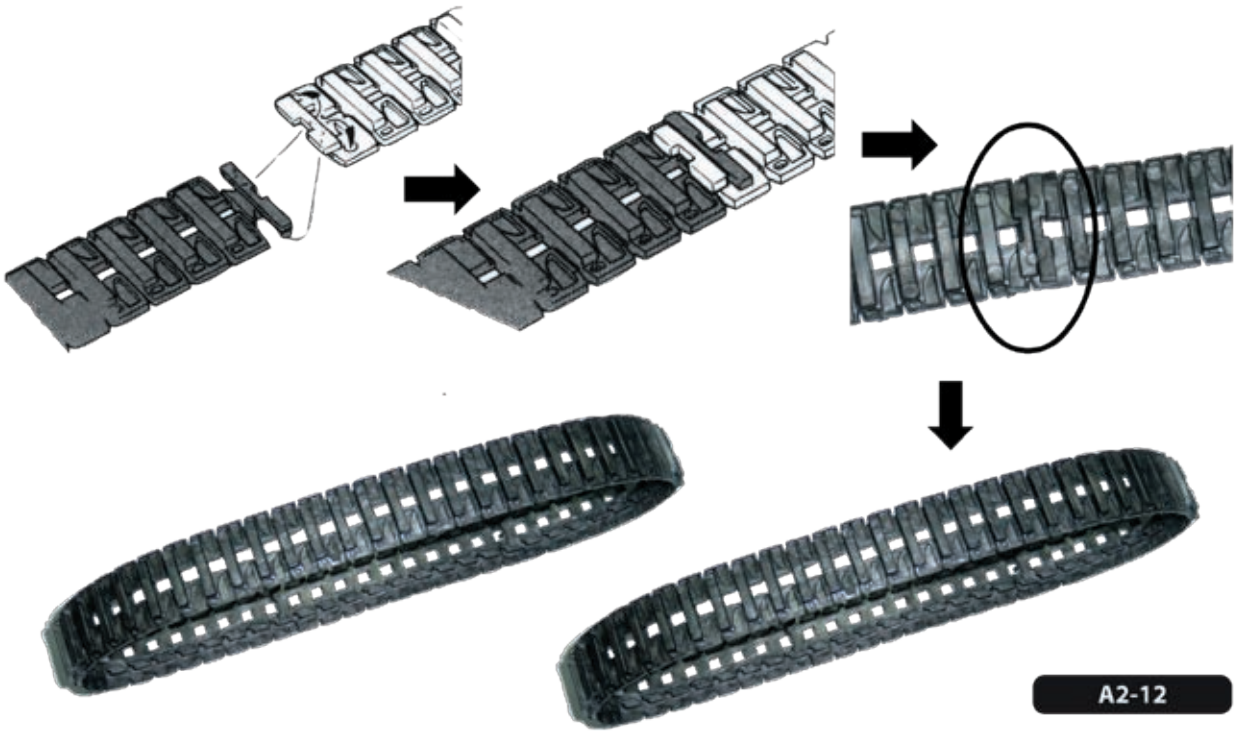
A2-10

A2.8 Оставьте конструкцию повернутой верхней стороной вниз. Вставьте металлические оси во второе и шестое отверстия длинной уголковой планки, как показано на рисунке A2-11. Наденьте средние направляющие колеса на металлические оси. Вставьте в колеса заглушки таким образом, чтобы колеса плотно сидели на осях. Переверните основание. Вставьте третью металлическую ось в отверстие короткой уголковой планки. Наденьте на ось большое направляющее колесо. Вставьте в колеса заглушки таким образом, чтобы колеса плотно сидели на осях.



A2-11

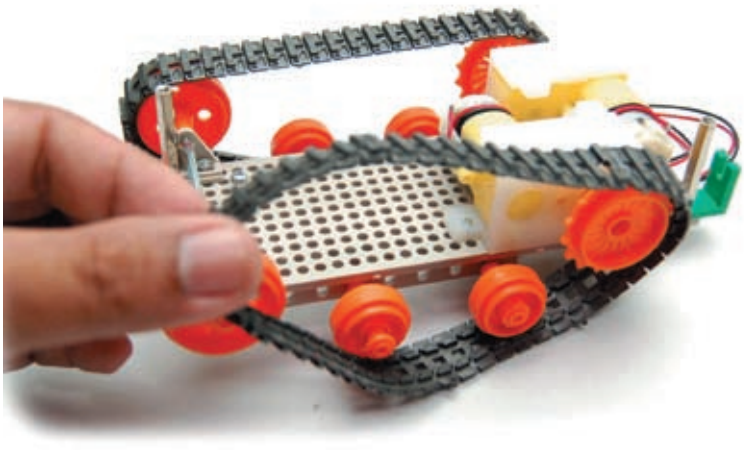
A2.9 Соберите две гусеницы, соединив вместе траки разного размера. Для изготовления одной необходимо: один трак с 30 звеньями и два трака с 10 звеньями. Соедините все три трака вместе. Возьмите один конец получившегося трака и соедините его с другим концом так, чтобы получилась замкнутая петля. Повторите описанные шаги для изготовления второй гусеницы. Если гусеницы одеваются на колеса слишком плотно или болтаются, можно попробовать изменить размер трака или изменить место крепления короткой уголковой планки к основанию до тех пор, пока гусеницы не станут двигаться легко и надежно.



A2-12

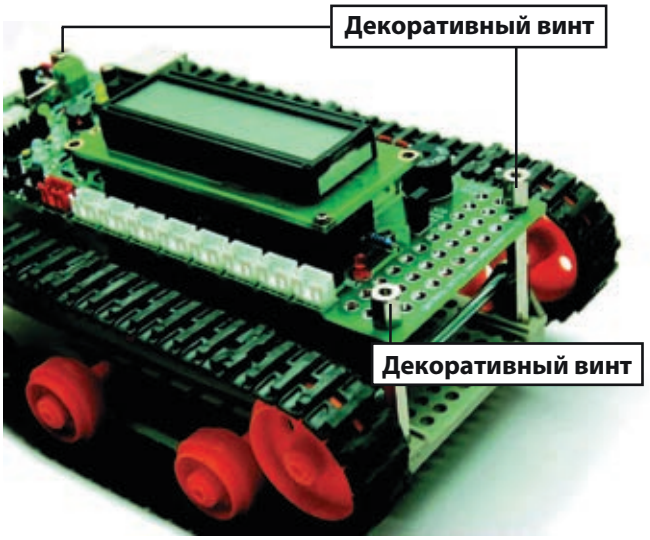
Пример, приведенный выше, является только примером, показывающим сборку гусеницы стандартной длины. Вы можете, конечно, собрать Вашу гусеницу другой длины, которая будет зависеть от Ваших собственных требований к изготавливаемому роботу

A2.10 Оденьте гусеницу на направляющие колеса робота.



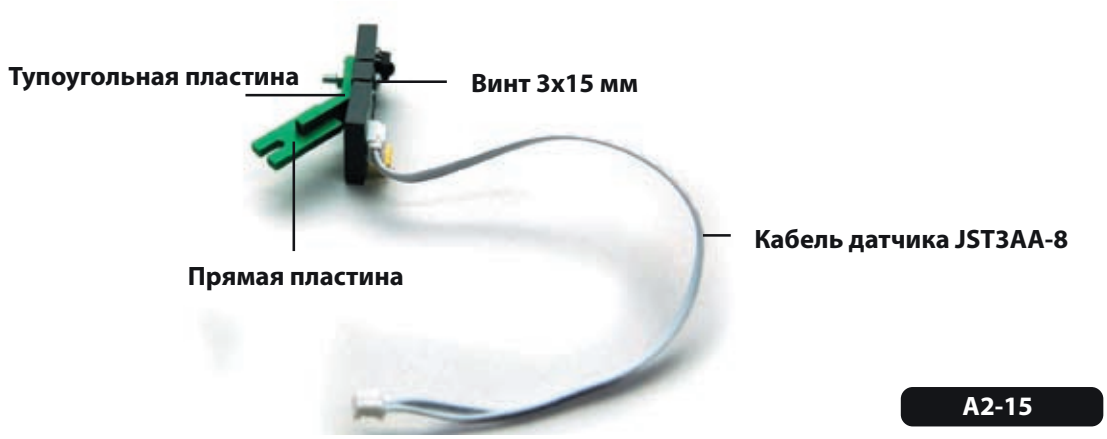
A2-13

A2.11 Присоедините плату управления RBX-877 V2.0 к верхней стороне корпуса робота. Закрепите, пожалуйста, плату таким образом, чтобы выключатель питания оказался со стороны расположения редукторов с двигателями постоянного тока. Закрепите плату тремя декоративными винтами по углам платы.



A2-14

A2.12 Присоедините плату датчика модуля приемника ZX-IRM 38kHz к тупоугольной пластине, используя винт 3x15 мм и гайку 3 мм. С другой стороны тупоугольной пластины вставьте прямую пластину.

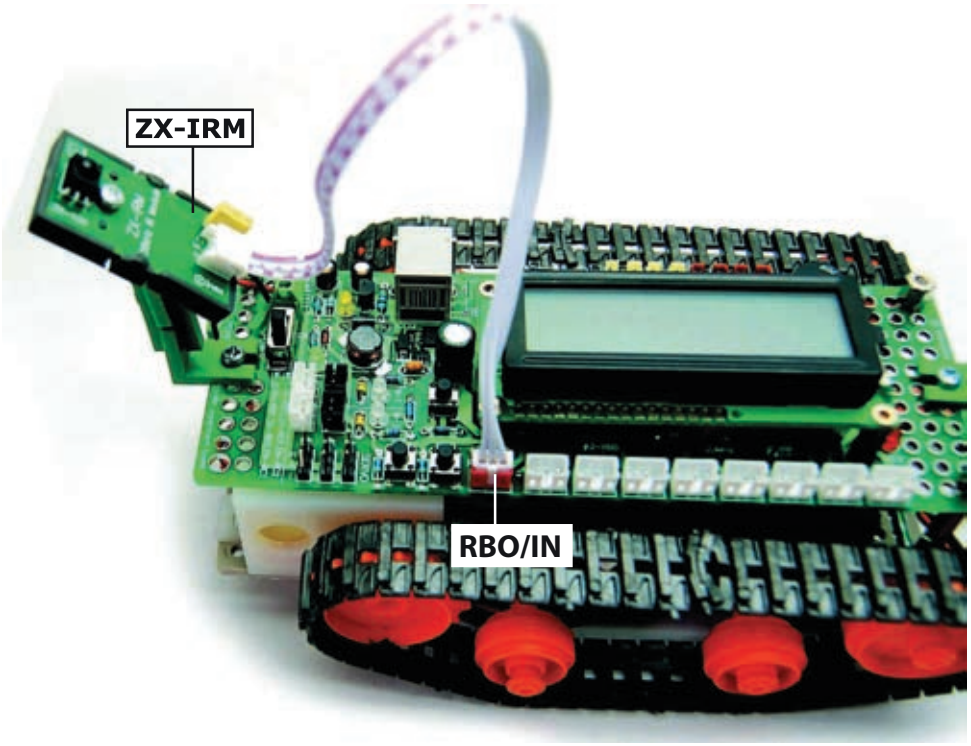


A2-15

A2.13 Для установки платы датчика ZX-IRM присоедините прямоугольную пластину к центральному отверстию с задней стороны (со стороны выключателя питания) платы управления RBX-877 V2.0 винтом 3x10 мм и гайкой 3 мм.

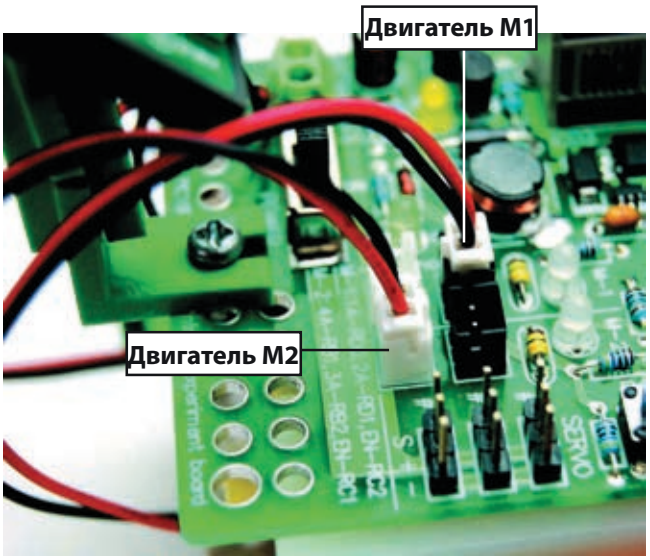
A2-16

A2.14 Присоедините модуль ZX-IRM, собранный на шаге A2.12, к прямоугольной пластине на плате управления RBX-877 V2.0, установленной на шаге A2.13. Вставьте кабель датчика Zx-IRM в разъем RB0/INT на плате управления RBX-877 V2.0.



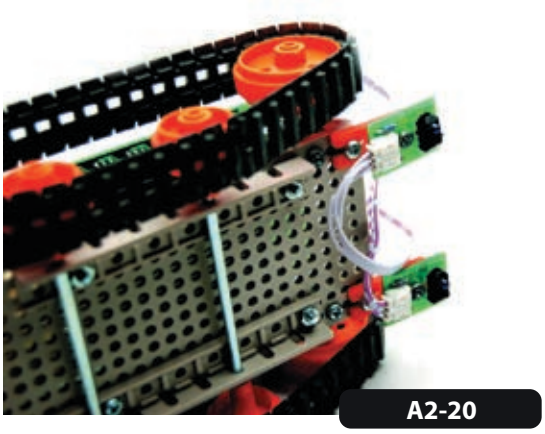
A2-17

A2.15 Вставьте кабель, идущий от коробки передач с двигателем постоянного тока, в разъем для подключения двигателя на плате управления RBX-877 V2.0. Правый двигатель необходимо подключить к белому разъему выхода M-2, левый двигатель необходимо подключить к черному разъему выхода M-1. Однако, полярность подключения двигателей можно изменить (белый или черный разъем) в зависимости от программы и назначения робота. В нормальной, относительно выходного индикатора двигателя, ситуации, если оба светодиода светятся зеленым светом, то это означает, что происходит движение вперед и, если оба светодиода светятся красным светом, то происходит движение назад. При некорректном выполнении какой-либо описанной выше операции, подключение двигателей и работу светодиодов можно изменить позже.



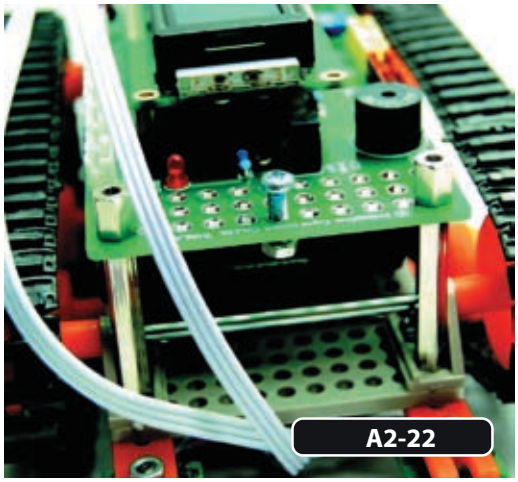
A2-18

A2.16 Установите плату инфракрасного отражательного датчика ZX-03 с нижней стороны корпуса робота. Присоедините датчик к крайнему отверстию плоской пластины с тремя отверстиями, пропустив винт 3x10 мм через плату датчика, 3-миллиметровой пластиковую шайбу, пластину и закрепив 3-миллиметровой гайкой. Установите обе платы. Одну – с правой стороны корпуса робота, другую – с левой стороны.

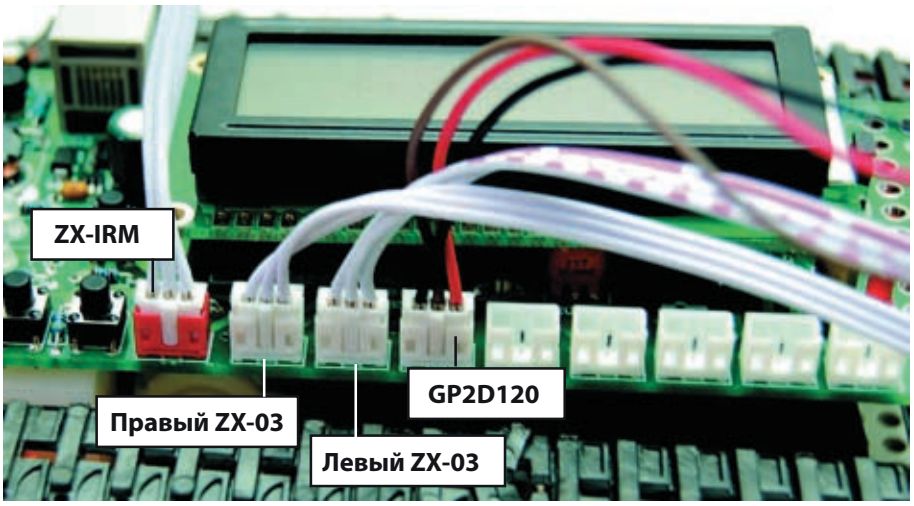


A2.17 Соедините модуль GP2D120 с прямоугольной пластиной как показано на Рисунке A2-21, используя винт 3x10 мм и гайку 3 мм.

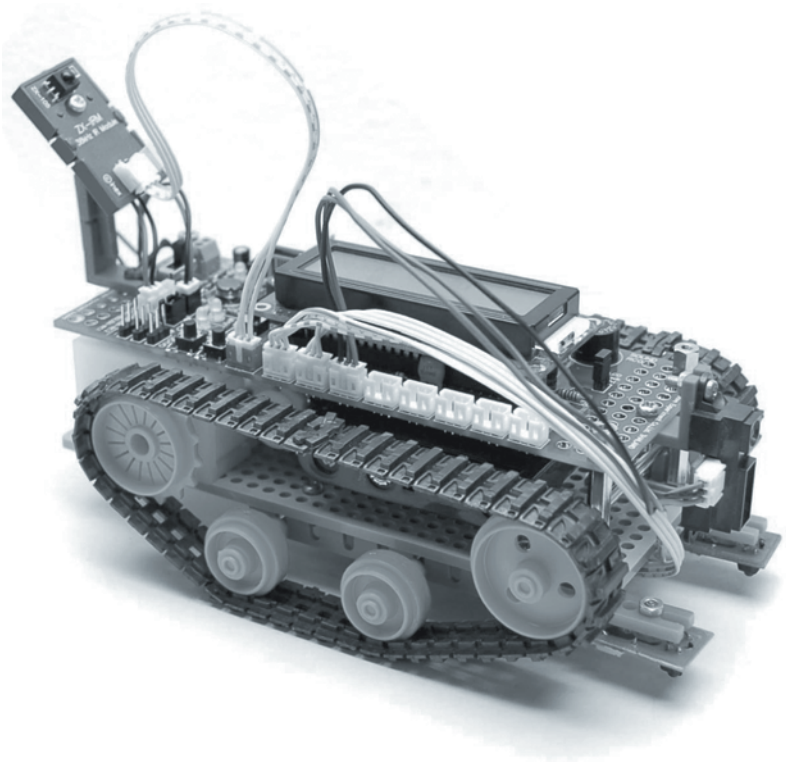
A2.18 Спереди робота, вставьте винт 3x10 мм в центральное отверстие платы RBX-877V2.0 и гайку 3 с верхней стороны платы, как показано на Рисунке A2-22. Не затягивайте винт. Далее, вставьте модуль GP2D120, собранный на шаге A2.17, между головкой винта и платой управления (см. Рисунок A2-23). Затяните винт для фиксации всего вместе.



A2.19 Вставьте кабель GP2D120 в разъем порта RA2, кабель левого датчика ZX-03 – в разъем порта RA0 и кабель правого датчика ZX-03 в разъем порта RA1.



A2.20 Подровняйте все кабели и проверьте правильность всех соединений. **Теперь Ваш Robo-PICA готов к программированию.**



Глава 4

Программное управление простейшим роботом

Вначале будет рассмотрено управление движением робота. Сердцем этого движения является схема управления двигателями постоянного тока. В Robo-PICA управление происходит двигателями постоянного тока коробок передач. На рисунке 4-1 показана схема управления двигателями постоянного тока. PIC16F887 использует шесть выводов порта для подключения микросхемы для управления двумя двигателями постоянного тока.

Механизм управления двигателями разделен на 4 типа:

1. Вал двигателя вращается по часовой стрелке.
2. Вал двигателя вращается против часовой стрелки.
3. Вал двигателя не вращается.
4. Вал двигателя заблокирован или заторможен.

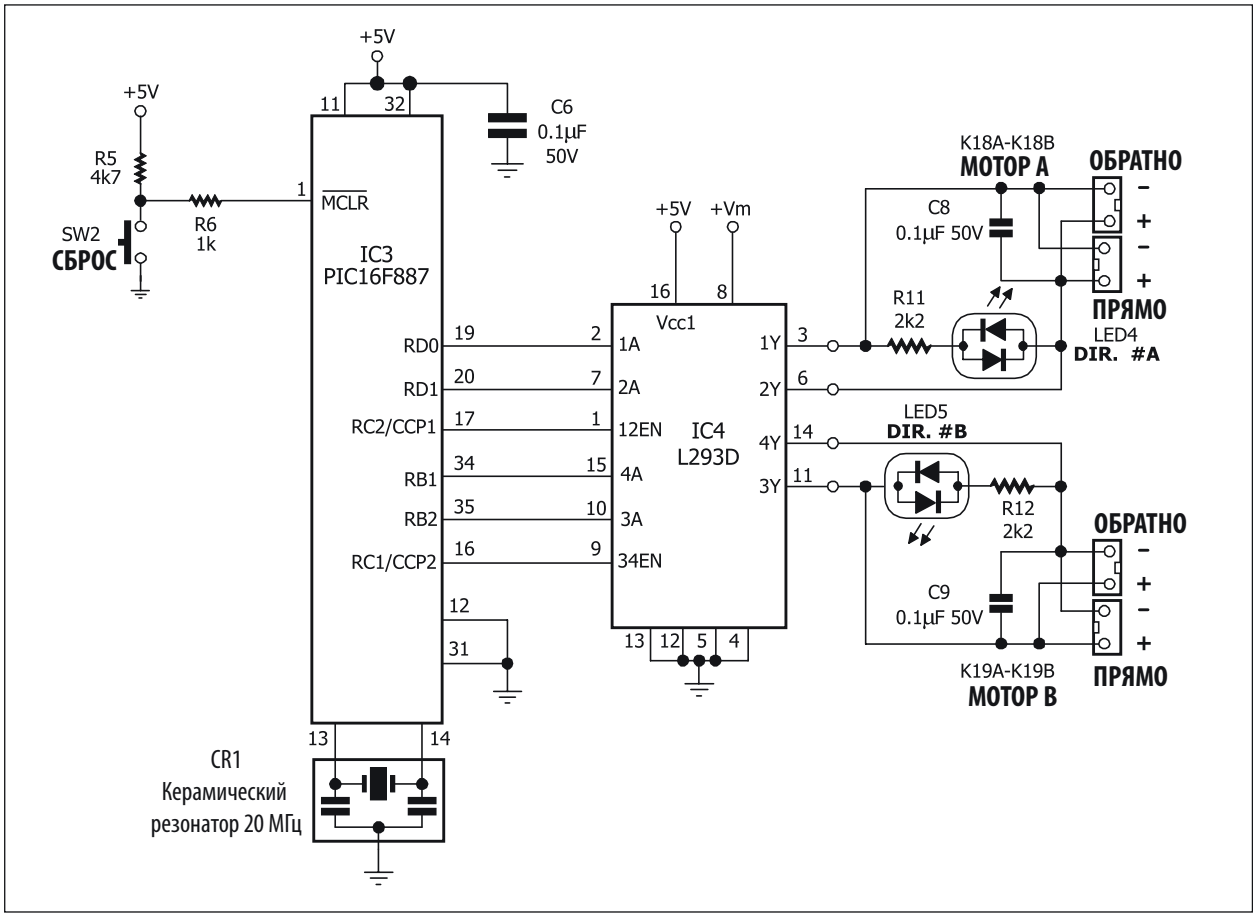


Рисунок 4-1: Схема модуля управления двигателем постоянного тока платы RBX-877 V2.0

Вывод 12EN/34EN	Вывод 1A/3A	Вывод 2A/4A	Функция работы двигателя
0	X	X	Двигатель обесточен
1	0	0	Блокировка привода или Торможение
1	0	1	Вращение по часовой стрелке
1	1	0	Вращение против часовой стрелки
1	1	1	Блокировка привода или Торможение

X означает любой логический уровень (либо "0", либо "1")

Таблица 4-1: Демонстрирует значения логических сигналов для управления направлением вращения двигателя

Сердцем блока управления двигателем постоянного тока является микросхема L293D полумостового драйвера (может быть заменена на SN754410). В Таблице 4-1 все сигналы, необходимые для контроля за блоком управления двигателя постоянного тока.

Выходы L293D присоединены к двигателям постоянного тока, расположенным в коробках передач, и их состояние отображается светодиодами, индицирующими полярность напряжения, подаваемого двигателем. Если напряжение подается в ПРЯМОЙ полярности, то светодиод светится Зеленым светом. И наоборот, если светодиод горит красным светом, это означает, что напряжение подается в ОБРАТНОЙ полярности. Разработчик, для индикации направления, может использовать другой цвет свечения светодиодов. Другими словами, если светодиоды светятся Красным светом, то робот будет двигаться назад, а если светодиоды светятся Зеленым светом, то робот будет двигаться вперед.

4.1 Файл библиотеки управления двигателем

Для улучшения производительности и удобства программирования, создана библиотека управления и контроля за движением для двигателей постоянного тока. Она располагается в файле ***motor.h***. Исходный текст этой библиотеки на языке C показан на Распечатке 4-1.

Чтобы создать эту библиотеку можно воспользоваться любым простейшим текстовым редактором и сохранить ее как .h-файл или открыть mikroC IDE, чтобы создать этот файл. После этого необходимо скопировать созданный файл в каталог библиотек компилятора mikroC, например, ***C:\Program Files\Mikroelektronika\mikroC\include***. Вы должны скопировать файл ***motor.h*** в этот каталог, поскольку компилятор будет использовать этот каталог для подключения любых файлов библиотек.

motor.h библиотечный файл содержит множество функций управления движением, включая:

Motor_Init: Инициализация выводов порта микроконтроллера для обмена с микросхемой драйвера двигателя постоянного тока.

Change_Duty: Управление скоростью вращения мотора.

Motor_A_FWD: Двигатель A (выход M-1) вращается в прямом направлении (светодиод, подключенный к разъему M-1, светится зеленым светом).

```
char motor_duty_= 127;                  // Ширина импульса ШИМ по умолчанию 50%
char motor_init_=0;                     // Начальное состояние
//      *** Motor A *****
//      PD0 ====> 1A
//      PD1 ====> 1B
//      PC2 ====> 1E (PWM1)
//      *** Motor B *****
//      PB1 ====> 2A
//      PB2 ====> 2B
//      PC1 ====> 2E (PWM2)
//*****************************************************
//********* Функция инициализации двигателя *****************
//*****************************************************
void Motor_Init()
   {
      if (motor_init_==0) // Это первый вызов функции ?
      {
         motor_init_=1;                 // Состояние
         ANSELH.F0=0;                   // RB1 ==> Цифровой Ввод/Вывод
         ANSELH.F2=0;                   // RB2 ==> Цифровой Ввод/Вывод
         TRISB.F1=0;                    // Двигатель Б 2A
         TRISB.F2=0;                    // Двигатель Б 2B
         TRISD.F0=0;                    // Двигатель A 1A
         TRISD.F1=0;                    // Двигатель A 1B
         Pwm1_Init(5000);               // Инициализация ШИМ1 => 1E
         Pwm2_Init(5000);               // Инициализация ШИМ2 => 2E
      }
}
//*****************************************************
//*****************************************************
//********* Управление шириной импульса ******************
//*****************************************************
void Change_Duty(char speed)
{
      if (speed != motor_duty_) // Получили то же значение скорости?
      {
         motor_duty_=speed;             // Сохранить текущее значение скорости
         Pwm1_Change_Duty(speed);       // Двигатель A
         Pwm2_Change_Duty(speed);       // Двигатель B
      }
}
//*****************************************************
/********** Двигатель A команда ВПЕРЕД ********/
void Motor_A_FWD()
{
      Pwm1_Start();
      PORTD.F0 =0;
      PORTD.F1 =1;
}
```

Распечатка 4-1: Исходный текст файла motor.h библиотеки управления двигателем постоянного тока (начало)

```
/***********************************/
/****** Двигатель B команда ВПЕРЕД ****/
void Motor_B_FWD()
{
        Pwm2_Start();
        PORTB.F1 =0;
        PORTB.F2 =1;
}
/***********************************/
/******** Двигатель A команда НАЗАД*****/
void Motor_A_BWD()
{
        Pwm1_Start();
        PORTD.F0 =1;
        PORTD.F1 =0;
}
/***********************************/
/******** Двигатель B команда НАЗАД *****/
void Motor_B_BWD()
{
        Pwm2_Start();
        PORTB.F1 =1;
        PORTB.F2 =0;
}
/***********************************/
/******* Двигатель A выключен **********/
void Motor_A_Off()
{
        Pwm1_Stop();
        PORTD.F0 =0;
        PORTD.F1 =0;
}
/***********************************/
/********** Двигатель B выключен *******/
void Motor_B_Off()
{
        Pwm2_Stop();
        PORTB.F1 =0;
        PORTB.F2 =0;
}
/***********************************/
/******* Команда ДВИЖЕНИЕ ВПЕРЕД ******/
void Forward(char speed)
{
        Motor_Init();
        Change_Duty(speed);
        Motor_A_FWD();
        Motor_B_FWD();
}
/***********************************/
```

Распечатка 4-1: Исходный текст файла motor.h библиотеки управления двигателем постоянного тока (продолжение)

```
/****** Команда ДВИЖЕНИЕ НАЗАД ********/
void Backward(char speed)
{
        Motor_Init();
        Change_Duty(speed);
        Motor_A_BWD();
        Motor_B_BWD();
}
/***********************************/
/***** Команда ПОВОРОТ НАПРАВО ** ******/
void S_Right(char speed)
{
        Motor_Init();
        Change_Duty(speed);
        Motor_A_FWD();
        Motor_B_BWD();
}
/***********************************/
/******** Команда ПОВОРОТ НАЛЕВО ******/
void S_Left(char speed)
{
        Motor_Init();
        Change_Duty(speed);
        Motor_A_BWD();
        Motor_B_FWD();
}
/***********************************/
/********** Команда СТОП ************/
void Motor_Stop()
{
        Motor_Init();
        Change_Duty(0);
        Motor_A_Off();
        Motor_B_Off();
}
/***********************************/
```

Распечатка 4-1: Исходный текст файла motor.h библиотеки управления двигателем постоянного тока (окончание)

Motor_B_FWD: Двигатель B (выход M-2) вращается в прямом направлении (светодиод, подключенный к разъему M-2, светится зеленым светом).

Motor_A_BWD: Двигатель A (выход M-1) вращается в обратном направлении (светодиод, подключенный к разъему M-1, светится красным светом).

Motor_B_BWD: Двигатель B (выход M-2) вращается в обратном направлении (светодиод, подключенный к разъему M-1, светится красным светом).

Motor_A_off: Выключение или Остановка двигателя A (выход M-1).

Motor_B_off: Выключение или Остановка двигателя B (выход M-2).

foward: Включены оба двигателя, обеспечивая движение Robo-PICA вперед.

backward: Включены оба двигателя, обеспечивая движение Robo-PICA назад.

S_right: Включены оба двигателя, обеспечивая поворот Robo-PICA направо.

S_left: Включены оба двигателя, обеспечивая поворот Robo-PICA налево.

Motor_stop: Остановлены оба двигателя.

Задание 3

Простейшее управление движением

Робот Robo-PICA двигается вперед или назад в том случае, если валы обеих коробок передач с двигателями постоянного тока одновременно вращаются в одном направлении. Если необходимо осуществить вращение или поворот, то методы выполнения этих операций описаны ниже:

1. **Остановка одного двигателя и Работа другого**. Если остановить левый двигатель и управлять правым двигателем, робот будет поворачивать налево. Для поворота в противоположном направлении, остановите правый мотор и управляйте левым мотором. Робот будет поворачивать влево. При этом скорость движения будет одинаковой. При этом поворотный момент будет приложен к точке неподвижной гусеницы. См. Рисунок А3-1.

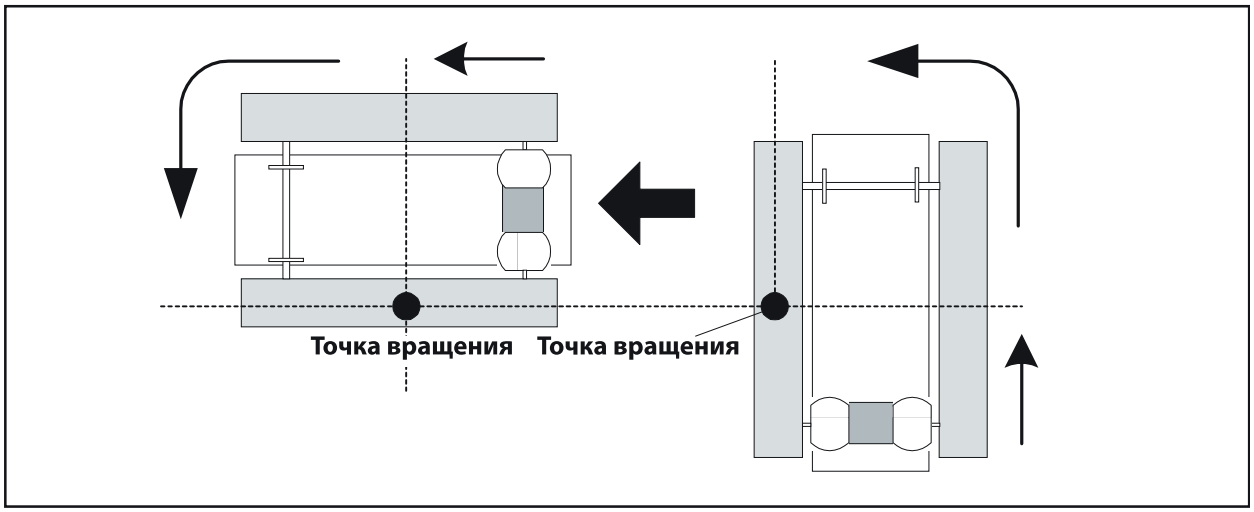


Рисунок А3-1: Вращение путем остановки двигателя и фиксации колес

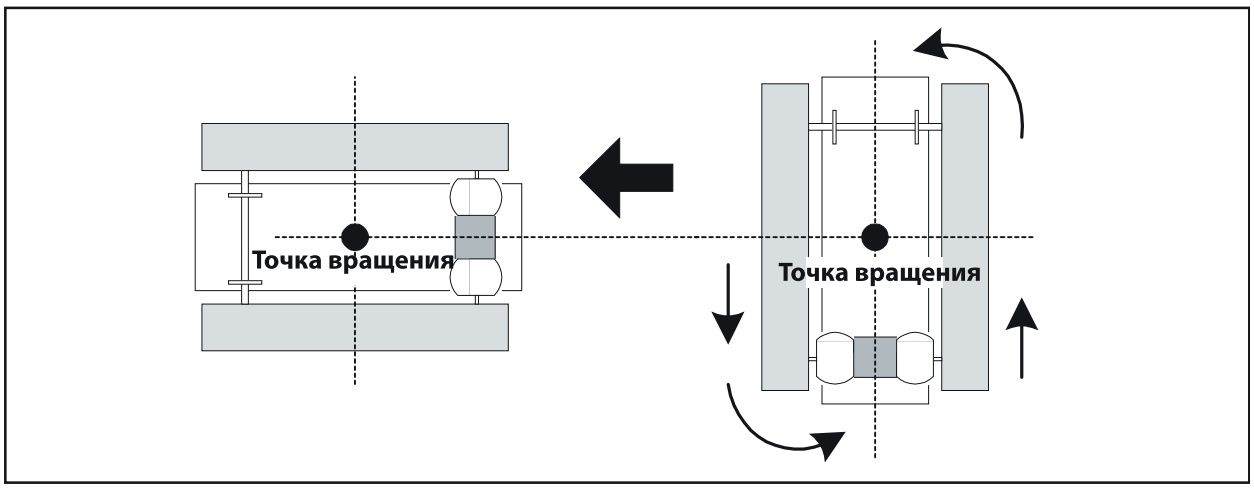


Рисунок А3-2: Вращение путем работы обоих двигателей в противоположных направлениях

2. **Работа обоих двигателей в противоположных направлениях**. Если левый двигатель будет вращаться вперед, а правый двигатель будет вращаться назад, робот будет поворачивать вправо. Если все будет происходить в обратную сторону, левый двигатель будет вращаться назад, а правый двигатель будет вращаться вперед, то робот вместо этого будет поворачивать влево. При использовании этого метода скорость поворота увеличивается в 2 раза, и значительно уменьшается сила трения. Вращение будет происходить относительно точки в центре робота. См. Рисунок A3-2.

A3.1 Наберите текст программы, следуя распечатке A3-1, затем скомпилируйте ее и загрузите в плату управления Роботом RBX-877 V2.0. Включите питание робота.

```
#include <motor.h>
void main()
{
   Sound_Init(&PORTC, 0);          // Инициализировать звуковой порт
   while(1)
   {
        Forward(255);              // Команда ВПЕРЕД
        Delay_ms(2000);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        S_Left(255);               // Команда ПОВОРОТ ВЛЕВО
        Delay_ms(800);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        Forward(255);              // Команда ВПЕРЕД
        Delay_ms(2000);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        S_Right(255);              // Команда ПОВОРОТ ВПРАВО
        Delay_ms(800);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        Forward(255);              // Команда ВПЕРЕД
        Delay_ms(2000);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        Backward(255);             // Команда НАЗАД
        Delay_ms(1000);
        sound_play(100,50);        // звуковой сигнал 1 кГц на выводе RC0
        Motor_Stop;                // Остановить оба двигателя
   }
}
```

Распечатка A3-1: Программа, иллюстрирующая управление движением Robo-PICA

A3.2 Отсоедините программирующий кабель от Robo-PICA. Поставьте робота на пол. Включите питание для выполнения программы. Наблюдайте за работой.

Робот будет двигаться вперед 2 секунды, поворачивать налево 0,8 секунды, двигаться вперед еще 2 секунды. Далее, он будет поворачивать направо 0,8 секунды, чтобы изменить направление движения, и двигаться вперед 2 секунды, двигаться назад 1 секунду и окончательно завершит движение. В каждый момент изменения типа движения робот будет подавать звуковой сигнал, чтобы сообщить о выполнении операции.

Возможно, однако, что робот будет двигаться в неправильном направлении. Если такое случилось, проверьте, пожалуйста, правильность подключения кабелей двигателей. Вы можете изменить подключение двигателя с белого на черный разъем и с черного на белый разъем для каждого выхода для подключения мотора.

Вы можете видеть цвет свечения светодиодов, подключенных к выходам для двигателей постоянного тока. Во время движения вперед оба светодиода должны светиться зеленым светом. Во время движения назад оба светодиода должны светиться красным светом. Следует изменять подключение двигателей до тех пор, пока направление движения не станет правильным и запомнить или зафиксировать правильные соединения для всех дальнейших Заданий.

Существует ограничение, накладываемое изготовителем двигателей. Невозможно точно узнать полярность подключения двигателей. Но ее можно контролировать и исправлять как на аппаратном, так и на программном уровне в схеме управления двигателем постоянного тока. Данная проблема может быть легко устранена и важно знать и понимать это.

Поскольку робот использует в качестве источника питания батареи, в процессе изменения заряда батарей от полного до неполного, скорость движения робота будет неодинаковой. В этом случае и проходимое роботом расстояние будет разным. Это ограничение присуще всем роботам, которые используют открытую петлю контроля за движением.

Задание 4

Управление скоростью движения Robo-PICA

Робот Robo-PICA может управлять скоростью движения, посылая сигнал на вход разрешения (EN) ИС управления двигателем L293D. Как показано на Рисунке A4-1 (в этой главе), вывод EN микросхемы L293D подключен к выводам порта RC2/CCP1 и RC1/CCP2 микроконтроллера PIC16F887. Оба вывода порта являются выходами порта ШИМ. Разработчик может написать программу для управления выходным сигналом ШИМ для регулирования частоты вращения двигателя.

Работа ШИМ

Стандартная техника управления двигателем предполагает подачу напряжения непосредственно на двигатель. При этом двигатель работает на полной скорости. Иногда эта скорость бывает слишком высокой. В таких случаях простейшим способом управления частотой вращения ротора мотора является регулировка подаваемого на него напряжения. Наиболее популярной является техника ШИМ (широтно-импульсная модуляция). При использовании этой техники происходит управление шириной положительного импульса. Напряжение, подаваемое на двигатель, будет вычисляться как среднее значение по периоду ШИМ. Отношение ширины положительного импульса к общей ширине импульса (периоду повторения импульса) называется коэффициентом заполнения. Он измеряется в процентах (%).

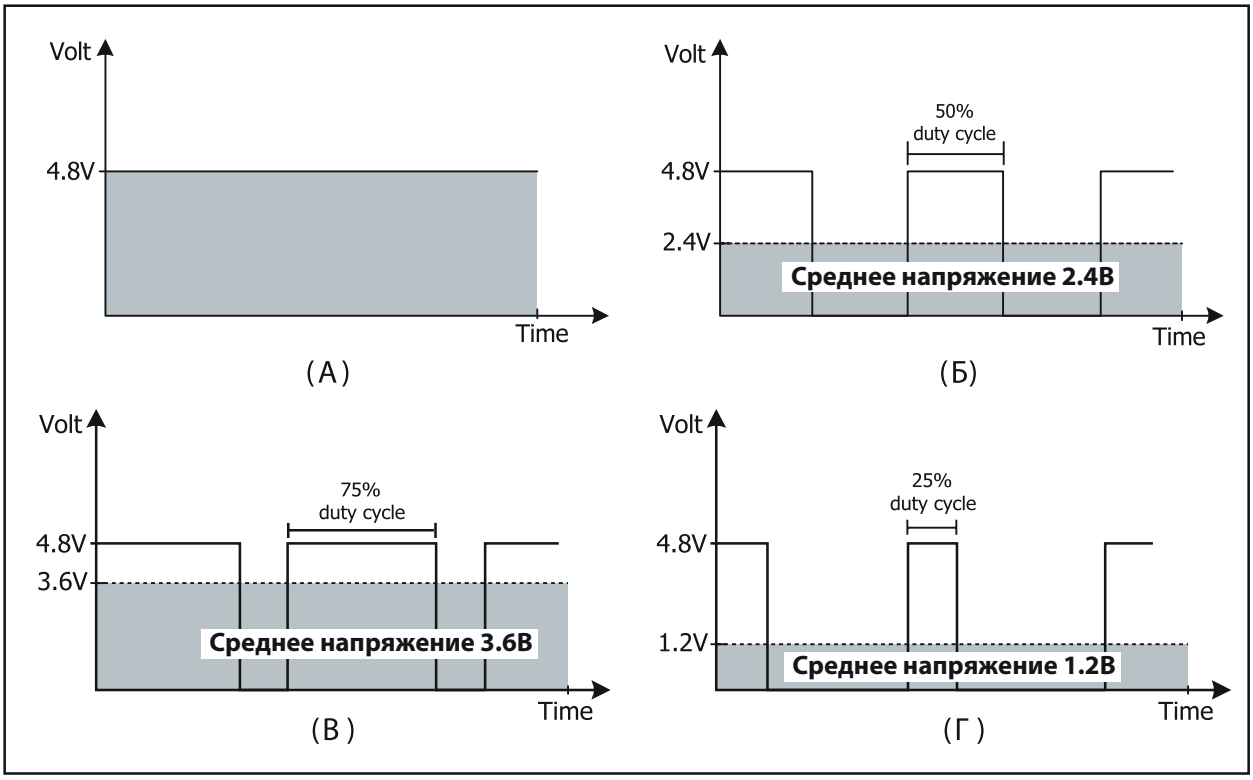


Рисунок A4-1: Показывает среднее значение напряжения на выходе ШИМ

(А) Полное напряжение
(Б) Коэффициент заполнения ШИМ 50%
(В) Коэффициент заполнения ШИМ 75%
(Г) Коэффициент заполнения ШИМ 25%

```
#include <motor.h>
char i;
void main()
{
        Forward(255);                          // Команда ВПЕРЕД
        while(1)
        {
                Delay_ms(2000);
                Pwm1_Change_Duty(220);         // Двигатель А 85% заполнение
                Pwm2_Change_Duty(255);         // Двигатель В 100% заполнение
                Delay_ms(5000);
                Pwm1_Change_Duty(255);         // Двигатель А 100% заполнение
        }
}
```

Распечатка А4-1: Программа управления скоростью движения Robo-PICA с использованием ШИМ

Для Robo-PICA есть уже готовая программа управления скоростью, основанная на технике ШИМ, расположенная в файле библиотеки ***motor.h***. Детали использования этой библиотеки можно увидеть в файле motor.h, исходный текст которого показан на Распечатке 4-1 (в этой главе). В библиотеку ***motor.h*** включены следующие функции ШИМ для поддержки 2 каналов ШИМ-микроконтроллера PIC16F887:

Pwm1_Change_Duty(speed); // Коэффициент заполнения двигателя А
Pwm2_Change_Duty(speed); // Коэффициент заполнения двигателя В

Вы можете подставить необходимое значение коэффициента заполнения в (speed). При изменении этого значения от 0 до 255 коэффициент заполнения меняется от 0 до 100%.

А4.1 Введите текст с Распечатки А4-1. Скомпилируйте и загрузите код в Robo-PICA. Выключите питание.

А4.2 Отсоедините загрузочный кабель от Robo-PICA.

А4.3 Положите робота на пол. Включите питание, чтобы выполнить программу. Наблюдайте за работой робота.

Робот Robo-PICA будет двигаться вперед с максимальной скоростью 2 секунды, и поворачивать влево 5 секунд. После этого робот вновь будет двигаться вперед с максимальной скоростью. Робот все время будет двигаться таким образом.

Наиболее подходящим значением коэффициента заполнения ШИМ для управления роботом является 70% и более. *Если выбрано меньшее значение, у робота может не оказаться вращающего момента, достаточного для вращений и поворотов.*

Глава 5

Бесконтактное обнаружение объектов

Одной из наиболее важных функций подвижного робота является обработка сигналов с датчиков. Робот Robo-PICA может обрабатывать сигналы с многих типов датчиков, поскольку он имеет как цифровые, так и аналоговые входы. Микросхема PIC16F887 – главный микроконтроллер Robo-PICA имеет много портов. В наборе используется 9 программируемых выводов портов для поддержки аналоговых и цифровых датчиков. Кроме того, используется 2 вида портов последовательного обмена данными; UART и шина I^2C.

В этой главе внимание будет сконцентрировано на взаимодействии с аналоговыми датчиками. В набор Robo-PICA входят 2 вида аналоговых датчиков: Инфракрасный Датчик Расстояния (дальномер) GP2D120 и Инфракрасный Отражательный Датчик ZX-03 для организации движения вдоль линий и контуров.

5.1 Аналого-цифровой преобразователь PIC16F887

Микроконтроллер PIC16F887 содержит модуль 14-канального 10-разрядного аналого-цифрового преобразователя (АЦП). Все порты аналогового ввода можно сконфигурировать, также, и для цифрового ввода/вывода. Они включают в себя RA0...RA3, RA5, RB0...RB5 и RE0...RE2.

Аналого-Цифровой Преобразователь (АЦП) позволяет преобразовывать аналоговый входной сигнал в его 10-разрядное двоичное представление. Этот модуль использует аналоговые входы, которые мультиплексируются в одну схему выборки-хранения. Выход схемы выборки-хранения соединен со входом преобразователя. Преобразователь выдает на выходе 10-разрядный двоичный результат Методом последовательных приближений и сохраняет результат преобразования в регистрах результата АЦП (ADRESL и ADRESH).

Опорное напряжение для АЦП можно выбрать программным путем между напряжением питания VDD или напряжением, подаваемым на вывод внешнего опорного напряжения.

5.2 Регистры АЦП

Важными для модуля АЦП являются регистры **ADCON0** и **ADCON1.** Регистр ADCON0 используется для функции выбора аналогового входа, а регистр ADCON1 используется для выбора формата выходных данных и источника образцового напряжения.

5.2.1 ADCON0: Регистр 0 Управления АЦП

Детальное описание каждого бита регистра ADCON0 показано ниже:

Бит	7	6	5	4	3	2	1	0
	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ DONE	ADON
	Ч/З-0	Ч/З-0	Ч/З-0	Ч/З-0	Ч/З-0	Ч/З-0	Ч/З-0	Ч/З-0

биты 7 и 6 – ADCS1, ADCS0: биты выбора частоты преобразования АЦП
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (частота, получаемая от специализированного внутреннего генератора = 500 кГц, максимум)

биты от 5 до 2 – CHS3 to CHS0: биты Выбора Аналогового Канала
0000 = AN0 (вывод RA0)
0001 = AN1 (вывод RA1)
0010 = AN2 (вывод RA2)
0011 = AN3 (вывод RA3)
0100 = AN4 (вывод RA5)
0101 = AN5 (вывод RE0)
0110 = AN6 (вывод RE1)
0111 = AN7 (вывод RE2)
1000 = AN8 (вывод RB2 – зарезервирован для схемы управления двигателем постоянного тока платы управления Роботом RBX-877V2.0)
1001 = AN9 (вывод RB3 – зарезервирован для светодиодного индикатора платы управления Роботом RBX-877V2.0)
1010 = AN10 (вывод RB1 – зарезервирован для схемы управления двигателем постоянного тока платы управления Роботом RBX-877V2.0)
1011 = AN11 (вывод RB4 – зарезервирован для выхода сервомотора платы управления Роботом RBX-877V2.0)
1100 = AN12 (вывод RB0 – альтернативная функция как вход внешнего прерывания и выключателя платы управления Роботом RBX-877V2.0)
1101 = AN13 (вывод RB5 pin – зарезервирован для выхода сервомотора платы управления Роботом RBX-877V2.0)
1110 = CVREF
1111 = Fixed Ref (фиксированное опорное напряжение 0,6 В)

бит 1 – GO/DONE: бит состояния АЦП

1 = цикл аналого-цифрового преобразования в процессе выполнения. Установка этого бита запускает цикл аналого-цифрового преобразования. Этот бит автоматически очищается микроконтроллером после окончания цикла аналого-цифрового преобразования.

"0" = цикл аналого-цифрового преобразования/не запущен

бит 0 – ADON: ADC бит разрешения работы АЦП

1 = работа АЦП разрешена

0 = работа АЦП, и он не потребляет электрический ток.

5.2.2 ADCON1: Регистр1 управления АЦП

Детальное описание каждого бита регистра ADCON1 показано ниже:

Бит	7	6	5	4	3	2	1	0
	ADFM	-	VCFG1	VCFG0	-	-	-	-
	Ч/З-0	Н-0	Ч/З-0	Ч/З-0	Н-0	Н-0	Н-0	Н-0

бит 7 – ADFM: бит выбора формата результата преобразования АЦП

1 = Правое выравнивание

0 = Левое выравнивание

бит 6 – Не задействован: всегда читается как "0"

бит 5 – VCFG1: бит выбора источника образцового напряжения

1 = вывод VREF

0 = напряжение питания Vss

бит 4 - VCFG0: бит выбора источника образцового напряжения

1 = вывод VREF+

0 = общий провод VDD

биты от 3 до 0 – Не задействованы: всегда читаются как "0"

5.2.3 ANSEL: регистр Выбора Аналогового Входа

Регистр ANSEL используется для конфигурирования входного режима линий ввода/вывода как аналоговых входов. Установка соответствующего бита ANSEL в состояние высокого уровня приводит к тому, что любое чтение цифрового состояния входа дает результат '0', и позволяет для этого входа корректно выполняться аналоговым функциям.

Детальное описание каждого бита регистра ANSEL показано ниже:.

Бит	7	6	5	4	3	2	1	0
	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1

биты от 7 до 0 – ANS7...ANS0: биты выбора аналогового входа

Выбор между аналоговыми и цифровыми функциями выводов AN<7:0> или RE2, RE1, RE0, RA5, RA3, RA2, RA1 и RA0 соответственно.

1 = Аналоговый вход. Вывод назначен как аналоговый вход (по умолчанию).

0 = Цифровой Вход/Выход. Вывод назначен как порт, или для специальных функций.

5.2.4 ANSELH: Старший байт регистра Выбора Аналогового Входа

Регистр ANSELH используется для конфигурирования входного режима линий ввода/вывода как аналоговых входов. Установка соответствующего бита ANSELH в состояние высокого уровня приводит к тому, что любое чтение цифрового состояния входа дает результат '0', и позволяет для этого входа корректно выполняться аналоговым функциям. Регистр управляет конфигурированием выводов порта AN8...AN13 (RB2, RB3, RB1, RB4, RB0 и RB5).

Детальное описание каждого бита регистра ANSELH показано ниже:

Бит	7	6	5	4	3	2	1	0
	-	-	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
	Н-0	Н-0	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1	Ч/З-1

биты 7 и 6 – Не задействованы: всегда читаются как "0"

биты от 5 до 0 – ANS13...ANS8: биты выбора аналогового входа

Выбор между аналоговыми и цифровыми функциями выводов AN<13:8> или RB5, RB0, RB4, RB1, RB3 и RB2 соответственно.

1 = Аналоговый вход. Вывод назначен как аналоговый вход (по умолчанию).

0 = Цифровой Вход/Выход. Вывод назначен как порт, или для специальных функций.

5.3 Конфигурирование АЦП

Для использования модуля АЦП микроконтроллера PIC16F887 необходимо рассмотреть следующие действия:

- Конфигурирование порта
- Выбор канала
- Выбор источника образцового напряжения для АЦП
- Выбор источника частоты преобразования
- Выбор способа форматирования результата преобразования

5.4 Конфигурирование порта

АЦП можно использовать для преобразования как аналоговых, так и цифровых сигналов. При преобразовании аналоговых сигналов, линии Ввода/Вывода необходимо сконфигурировать как аналоговые входы, установив соответствующие биты регистров TRIS и ANSEL.

5.4.1 Выбор канала

Биты CHS регистра ADCON0 определяют какой из каналов будет подключен ко входу схемы выборки-хранения. После выбора канала, перед выполнением следующего преобразования, требуется некоторая задержка.

5.4.2 Выбор источника образцового напряжения для АЦП

Биты VCFG регистра ADCON0 обеспечивают независимое управление источниками положительного и отрицательного опорного напряжения. В качестве источника положительного опорного напряжения можно использовать либо напряжение источника питания Vdd, либо внешний источник стабильного напряжения. Точно так же, в качестве источника отрицательного опорного напряжения, можно использовать напряжение источника питания Vss, либо внешний источник стабильного напряжения.

Для платы управления Роботом RBX-877V2.0 будет выбрано напряжение питания +5 В в качестве источника положительного опорного напряжения и напряжение Vss или общая шина устройства в качестве источника отрицательного опорного напряжения.

5.4.3 Выбор источника частоты преобразования

Источник частоты преобразования выбирается программно посредством бита ADCS регистра ADCON0. Существует 4 варианта такого выбора:

- $F_{OSC}/2$: для тактовой частоты 20 МГц, T_{AD} = 100 нсек
- $F_{OSC}/8$: для тактовой частоты 20 МГц, T_{AD} = 400 нсек
- $F_{OSC}/32$: для тактовой частоты 20 МГц, T_{AD} = 1,6 мксек
- F_{RC} (специальный внутренний генератор): T_{AD} = 2...6 мксек

Время, необходимое для получения одного разряда преобразования, обозначено как T_{AD}. На один полный цикл 10-разрядного преобразования требуется 11 T_{AD} периодов.

5.4.4 Форматирование результата преобразования

Результат аналого-цифрового преобразования сохраняется в регистровой паре ADRESH:ADRESL. Форма размещения результата преобразования будет зависеть от установки бита ADFM.

Если выбрано **левое** выравнивание (ADFM = '0'), регистр ADRESH будет содержать 8 старших битов и регистр ADRESL – оставшиеся 2 младших.

Если выбрано **правое** выравнивание (ADFM = '1'), регистр ADRESH будет содержать 2 старших бита и регистр ADRESL – оставшиеся 8 младших битов.

Формат результирующих данных в зависимости от выбора значения бита ADFM регистра ADCON1

5.5 Процедура Аналого-Цифрового преобразования

Рассмотрим пример процедуры использования АЦП для выполнения аналого-цифрового преобразования микроконтроллером PIC16F887:

1. Конфигурирование порта:
   - Отключить выходной драйвер вывода
   - Сконфигурировать вывод как аналоговый вход, записав необходимое значение в регистр ANSEL или регистр ANSELH
2. Конфигурирование модуля АЦП (используя регистр ADCON0):
   - Выбрать частоту преобразования АЦП
   - Сконфигурировать источник опорного напряжения
   - Выбрать входной канал АЦП
   - Выбрать формат результата преобразования, используя регистр ADCON1
   - Включить модуль АЦП
3. Конфигурирование Прерываний от АЦП (не обязательно):
   - Очистить флаг прерывания от АЦП
   - Разрешить прерывания от АЦП
   - Разрешить прерывания от периферии
   - Разрешить глобальные прерывания.

4. Ожидание окончания завершения всех вышеперечисленных операций в течение некоторого промежутка времени.

5. Запуск преобразования установкой бита GO/DONE регистра ADCON0.

6. Ожидание завершения АЦП-цикла преобразования одним из следующих способов:
   - Опрос бита GO/DONE
   - Ожидание прерывания от АЦП (прерывания должны быть разрешены)

7. Чтение результата преобразования АЦП. Данные результата преобразования записываются в регистры ADRESH и ADRESL.

8. Очистка флага прерывания от АЦП (требуется, если прерывания разрешены).

5.6 GP2D120: Инфракрасный дальномер (датчик расстояния) для дистанций 4...30 см.

Одним из специальных датчиков в робототехнике является **инфракрасный датчик расстояния**. Некоторый люди называют его ИК-локатор. Использование модуля GP2D120 добавляет роботу возможность измерения расстояний или обнаружения препятствий, используя инфракрасное излучение. Робот Robo-PICA может обходить препятствия без необходимости какого-либо физического контакта с ними.

5.6.1 Возможности GP2D120

Для измерения расстояний используется отражение инфракрасного излучения.

Измеряет расстояния от 4 до 30 см.

Требует напряжения питания от 4,5 до 5 В при потребляемом токе 33 мА.

Выходное напряжение находится в диапазоне от 0,4 до 2,4 В, при напряжении питания +5 В.

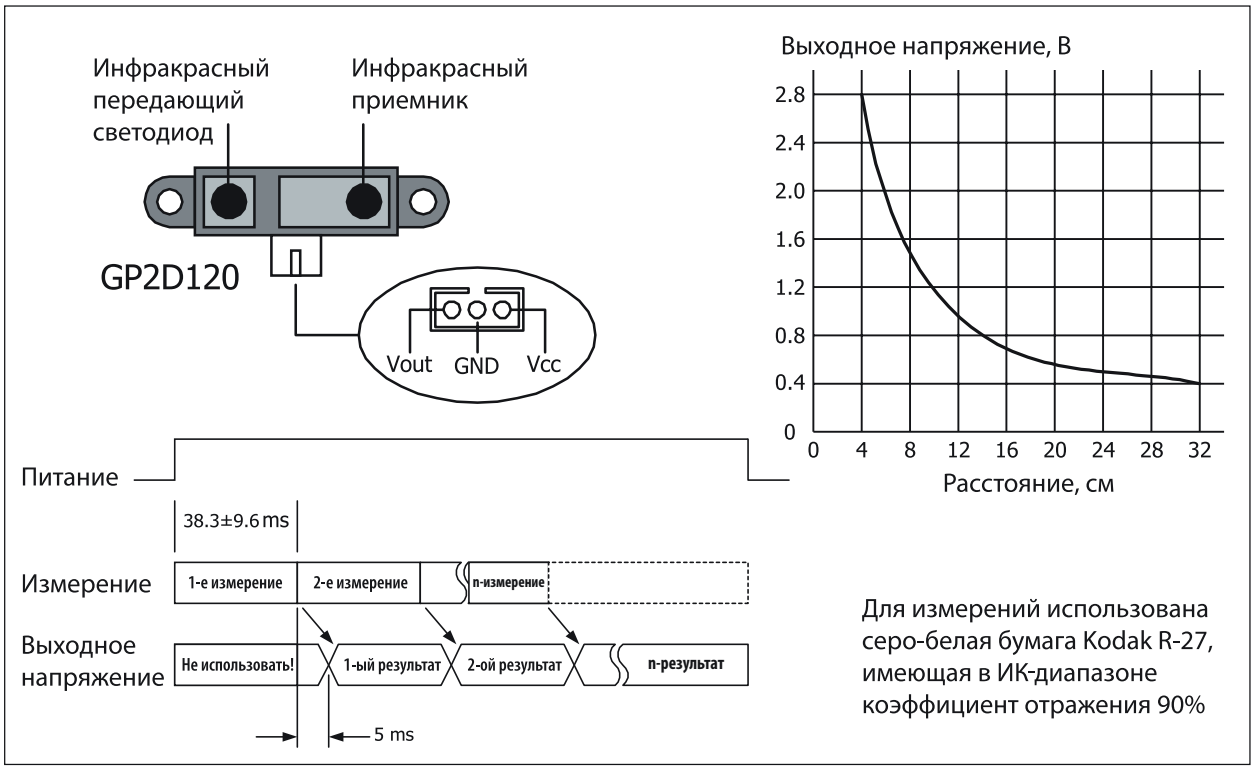


Рисунок 5-1: Расположение выводов GP2D120 и его рабочие характеристики

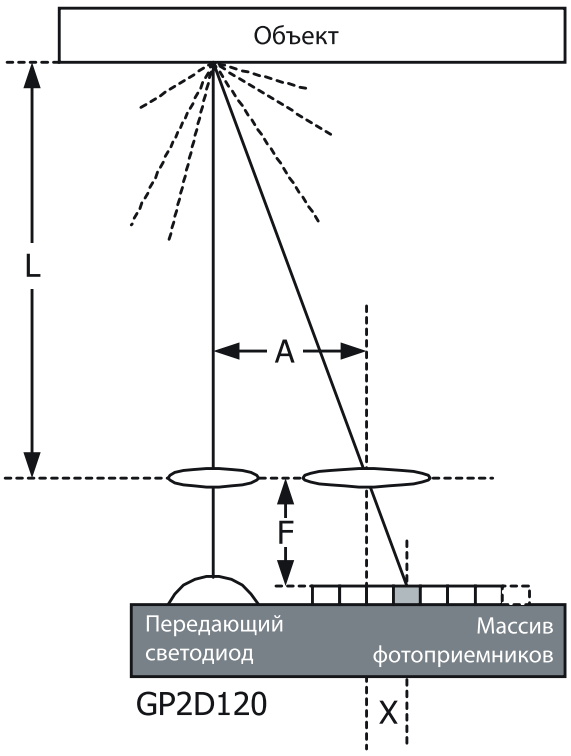
Модуль Инфракрасного Локатора GP2D120 имеет 3 вывода: Вход напряжения питания (Vcc), Земля (GND), Выходное напряжение (Vout). Чтобы прочитать значение выходного напряжения с модуля GP2D120, необходимо подождать в течение промежутка времени, необходимого для проведения измерения, который составляет примерно 32...52,9 мсек.

Выходное напряжение GP2D120 при измерении расстояния 30 см и напряжении питания +5 В находится между 0,25 и 0,55 В, со средним значением 0,4 В. При измерении расстояния 4 см, выходное напряжение будет иметь значение 2,25± 0,3 В.

5.6.2 Принцип работы модуля ИК-локатора

Измерение расстояний можно проводить множеством разных способов. Самым легким для понимания является измерение с использованием ультразвука, при котором звуковые волны посылаются на объект, и измеряется время, прошедшее с момента отсылки сигнала до момента приема отраженного сигнала. Поскольку скорость звуковых волн не очень большая, такие измерения можно проводить, используя несложное оборудование. Однако, в случае инфракрасного излучения, время, прошедшее с момента отсылки светового сигнала и до момента приема отраженного от препятствия нельзя измерить простыми способами, поскольку скорость распространения инфракрасного излучения очень велика (равна скорости света). Оборудование для таких измерений труднодоступно. Поэтому следует использовать следующую теорию.

Инфракрасное излучение посылается передатчиком на объект впереди, проходя через собирающую линзу, которая фокусирует излучение на центральную точку. Отражаясь от объекта, свет преломляется. Часть преломленного света попадет обратно, на приемник, в котором другая линза объединит лучи и определит точку попадания света на приемник, представляющий собой линейку фототранзисторов. Положение фототранзистора, на который упадет максимальное количество света, будет использовано для определения расстояния (L) от передатчика до препятствия по следующей формуле:



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

откуда L равно

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

Таким образом, значение расстояния от фототранзисторов, перед его преобразованием в напряжение, будет отправлено в Модуль Обработки Сигнала, приводя к изменениям выходного напряжения в соответствии с измеренным расстоянием.

5.6.3 Чтение сигнала GP2D120 с использованием АЦП

Выходное напряжение GP2D120 будет изменяться согласно измеренному расстоянию. Например, выходному напряжению 0,5 В соответствует расстояние 26 см и выходному напряжению 2 В соответствует расстояние 6 см. На диаграмме на Рисунке 5-1 показана зависимость выходного напряжения от расстояния для GP2D120.

При соединении GP2D120 с модулем АЦП-микропроцессора, в качестве результата будем иметь сырые данные после аналого-цифрового преобразования. Пользователь должен использовать программу для преобразования сырых данных в соответствующее им расстояние. Приближенно, расстояние можно вычислить по приведенной ниже формуле:

$$R = \frac{2914}{V + 5} - 1,$$

где R – расстояние в сантиметрах
V – цифровое значение, полученное от АЦП

Например, (см. Рисунок 5-1).
Сырые данные, полученные от АЦП – 307.
Эквивалентное расстояние – 8 см.

Предупреждение относительно сигнального кабеля GP2D120

Модуль GP2D120 имеет назначение выводов, отличающееся от назначения выводов модуля, используемого в наборе MicroCamp, несмотря на то, что выглядят они одинаково. Поэтому, модуль GP2D120 в данном наборе поставляется с уже присоединенным к нему специальным кабелем. Пользователю достаточно присоединить другой конец этого кабеля к разъему на плате MicroCamp. Не вытаскивайте кабель из модуля, и не заменяйте его на сигнальные кабели от других модулей датчиков.

Выходное напряжение GP2D120, В	Результат преобразования 10-битного АЦП	Дистанция, см
0,4	82	32
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 5-1: Зависимость выходного напряжения GP2D120, показаний АЦП и измеряемой дистанции

Задание 5

Чтение аналогового сигнала

Это задание описывает простейший эксперимент, поясняющий основные принципы чтения аналоговых сигналов. В качестве источника аналогового сигнала используется простой Переменный резистор или Потенциометр и присоединяется к аналоговому входному порту микроконтроллера PIC16F887. Для чтения результата с модуля АЦП в микроконтроллер PIC16F887 и отображения на ЖКИ-индикаторе пишется простая программа на языке С.

A5.1 Введите текст, показанный на Распечатке A5-1. Скомпилируйте и загрузите полученный код в плату управления Роботом RBX-877 V2.0.

A5.2 Присоедините потенциометрический датчик ZX-POTH к разъему порта RA3-платы управления Роботом RBX-877 V2.0.

A5.3 Запустите программу. Вращайте ручку потенциометра и наблюдайте за результатом на экране ЖКИ-индикатора платы управления Роботом RBX-877 V2.0.

Модуль ЖКИ отображает сообщение **SENSOR1 = xxx** *(***xxx** *изменяется от 0 до 1023).*

```
/************************************/
/** Отображает данные АЦП с RA1 на ЖКИ */
/************************************/
char data_[6];
int x;
void main()
{
   Delay_ms(1000);
   Lcd_Init(&PORTD);
   ANSEL = 0xFF;                    // PORTA ==> Аналоговый
   TRISA = 0xFF;                    // PORTA ==> вход
   Lcd_Cmd(LCD_CURSOR_OFF);         // Курсор ЖКИ выключен
   Lcd_Out(1,1,"SENSOR1 = ");       // Показать текст
   ADCON0=0b11001101;               // Выбор режима: Analog1 RC_Mode и ADON
   while(1)
   {
        ADCON0.GO=1;
        while(ADCON0.GO);
        x= (ADRESH*4)+(ADRESL/64);
        WordToStr(x,data_);
        Lcd_Out(1,10,data_);
        Delay_ms(100);
   }
}
```

Распечатка A5-1: Программа, демонстрирующая чтение модулем АЦП сигнала с датчика

Задание 6

Изучение GP2D120

A6.1 Введите текст, показанный на распечатке A6-1. Скомпилируйте и загрузите полученный код в плату Управления Роботом RBX-877 V2.0.

A6.2 Подключите датчик GP2D120 к разъему порта RA2-платы Robo-PICA. (Вы это уже делали в Задании 3).

A6.3 Запустите программу. Расположите какой-нибудь предмет перед модулем датчика GP2D120. Наблюдайте за показаниями ЖКИ.

A6.4 Изменяйте расстояние от объекта до датчика GP2D120 и наблюдайте за результатом.

В результате проверки Вы увидите, что датчик GP2D120 может обнаруживать предметы на расстоянии от 4 до 30 см.

```
int Adc;
char txt[6];
void Read_Adc()
{
   ADCON0=0b11001001;                  // Выбор Analog2 RC_Mode и ADON
   ADCON0.GO=1;                        // Запуск преобразования
   while(ADCON0.GO);                   // Ожидаем, пока преобразование завершится
   Adc=(ADRESH*4)+(ADRESL/64);         // 10 бит Данных ==> Adc
}
void main()
{
   Delay_ms(1000);
   Lcd_Init(&PORTD); // Инициализация ЖКИ
   Lcd_Cmd(LCD_CURSOR_OFF); // Курсор ЖКИ выключен
   Lcd_Out(1,1,"Raw Data= "); // Вывод текста первой строки
   while(1)
   {
        Read_Adc();
        WordToStr(Adc,txt);            // Отображение результата преобразования на ЖКИ
        Lcd_Out(1,10,txt);
        if (Adc<90)                    // Если данные < 90 – то они за пределами области допустимого
        {
              Lcd_Out(2,1,"Out of Range");
        }
        else
        {
              Adc = (2914/(Adc+5))-1;  // Преобразование Данных в сантиметры
              WordToStr(Adc,txt);      // Преобразование Данных в строку
              Lcd_Out(2,1,"In CM= ");  // Отображение результата на ЖКИ
              Lcd_Out(2,6,txt);
        }
        Delay_ms(1000);
   }
}
```

Распечатка A6-1: Пример работы GP2D120 с Robo-PICA

Задание 7

Бесконтактное обнаружение объектов роботом

A7.1 Введите текст, показанный на Распечатке A7-1. Скомпилируйте и загрузите полученный код в плату управления Роботом RBX-877 V2.0.

A7.2 Отключите питание и отсоедините от Robo-PICA загрузочный кабель.

```
/*********************************************************/
/***** Робот с Обнаружителем Объектов *********************/
/*********************************************************/
#include <motor.h>
int Adc;     // Хранит Аналоговые Данные
char Txt[6];                              // Хранит Строку
void Read_Adc()
{
   ADCON0=0b11011101;                     // Выбор Analog2 RC_Mode и ADON
   ADCON0.GO=1;                           // Запуск преобразования
   while(ADCON0.GO);                      // Ожидаем, пока преобразование завершится
   Adc=(ADRESH*4)+(ADRESL/64);            // 10 бит Данных ==> Adc
   }
void main()
{
   Delay_ms(1000);                        // Начальная задержка
   ANSELH.F4=0;                           // RB0 ==> Цифровой Ввод/Вывод
   ANSEL=0xFF;
   TRISA=0xFF;
   Lcd_Init(&PORTD);                      // Инициализация ЖКИ
   Lcd_Cmd(LCD_CURSOR_OFF);               // Курсор ЖКИ выключен
   while(PORTB.F0);                       // Ожидание нажатия кнопки
   while(1)
   {
        Read_Adc();                       // Чтение сигнала с Analog2
        WordToStr(Adc,Txt);               // Преобразование Данных в Строку
        Lcd_Out(1,1,Txt);                 // Отображение на ЖКИ
        if (Adc>300)                      // Если обнаружен объект в допустимом диапазоне расстояний
        {
             Backward(255);Delay_ms(500);  // Движение Назад и поворот Влево
             S_left(255);Delay_ms(400);
        }
        else
        {
             Forward(255); // Объект вне диапазона ВПЕРЕД
        }
   }
}
/*********************************************************/
```

Распечатка A7-1: Программа на языке C для бесконтактного обнаружения объектов роботом

А7.3 Положите робота на пол. Включите питание и наблюдайте за его работой.

А7.4 Попытайтесь располагать разные предметы перед роботом и наблюдайте за его реакцией.

Робот будет определять расстояние до объекта в диапазоне от 8 до 30 см. При отсутствии препятствий, робот будет непрерывно двигаться вперед. При обнаружении предмета, он будет двигаться назад, поворачивать налево и снова двигаться вперед.

Глава 6

Задача движения вдоль линии

Отслеживание линий или следование вдоль линии – популярная задача при любой Робототехнической деятельности. Целью этой главы является изучение взаимодействия с аналоговым датчиком. В робототехническом наборе Robo-PICA, это проще всего сделать, используя для этой задачи пару Инфракрасных отражательных датчиков. Добавим органы чувств к Robo-PICA так, что он сможет обнаруживать линии и двигаться вдоль них, используя Инфракрасный Отражательный Датчик. С передней стороны Robo-PICA будут установлены два Инфракрасных Отражательных Датчика так, что он сможет обнаруживать как белые, так и черные линии.

6.1 Инфракрасный Отражатель ZX-03

Сердцем этого датчика является оптопара с открытым оптическим каналом TCRT5000. Он создан для обнаружения с помощью инфракрасного (ИК) излучения объектов, находящихся в непосредственной близости. Он состоит из инфракрасного светодиода, закрытого синим, прозрачным для ИК-лучей, колпаком, и инфракрасного фототранзистора, закрытого черным колпаком. Когда инфракрасное излучение, испускаемое светодиодом, отражается от поверхности и возвращается на черное окно фототранзистора, попадая на его базу, то возникает ток проводимости. Чем больше излучения попадает на базу фототранзистора, тем больше его ток проводимости.

При использовании в качестве аналогового датчика, ZX-03 может обнаруживать тени или листы серой бумаги и, если освещенность в комнате постоянна, измерять расстояния в небольшом диапазоне.

Рабочий диапазон измерения расстояния до линии или пола составляет от 3 до 8 мм. Выходное напряжение датчика изменяется от 0,1 до 4,8 В, что соответствует диапазону значений на выходе 10-разрядного аналого-цифрового преобразователя от 20 до 1000. Таким образом, ZX-03 будет подходящим для использования его в качестве датчика отслеживания линий.

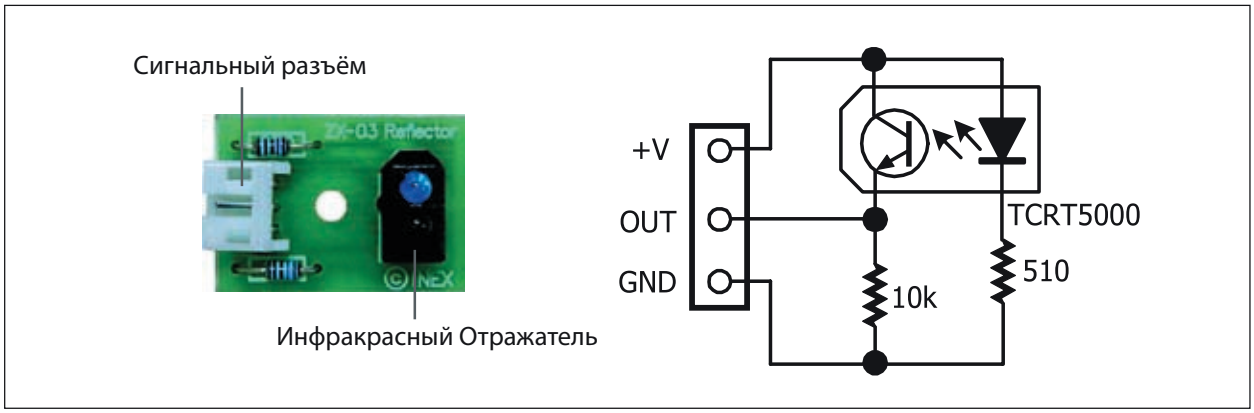


Рисунок 6-1: Информация об Инфракрасном Отражательном датчике ZX-03

Задание 8

Чтение датчика отслеживания линий

В собранном в Задании 2 Роботе Robo-PICA оба Инфракрасных отражателя ZX-03 установлены снизу корпуса робота и присоединены кабелями датчиков к выводам портов RA0 и RA1. В этом задании будет написана программа для чтения данных с датчиков ZX-03 и отображения их на ЖКИ-индикаторе робота Robo-PICA.

A8.1 Наберите программу, показанную на Распечатке A8-1. Скомпилируйте и загрузите код в Robo-PICA.

A8.2 Выключите питание и отсоедините от Robo-PICA загрузочный кабель.

A8.3 Поместите Robo-PICA на белую поверхность бумажного демонстрационного поля, которое входит в комплект Robo-PICA. Оба датчика ZX-03 должны располагаться над белой поверхностью. Включите питание. Посмотрите результат на экране ЖКИ и запишите его. Выключите питание.

A8.4 Поместите Robo-PICA на черную линию бумажного демонстрационного поля . Оба датчика ZX-03 должны располагаться над черной линией. Включите питание. Посмотрите результат на экране ЖКИ и запишите его.

При проверке может получиться следующий результат:

Значение, прочитанное над белой поверхностью, от 400 до 900
Значение, прочитанное над черной линией, от 0 до 150

Таким образом, пороговое значение может находиться в диапазоне от 150 до 400. Вы можете выбрать любое подходящее значение и принимать решение:

Если с датчика прочитано значение большее порогового, то обнаружена

“БЕЛАЯ поверхность”

Если с датчика прочитано значение меньшее порогового, то обнаружена
“ЧЕРНАЯ поверхность или линия”

```
/*******************************************************/
/************ Чтение датчика отслеживания линий **************/
/*******************************************************/
int Sensor0,Sensor1;                    // Хранит Аналоговое значение
char Txt[6];                            // Хранит результат преобразования в Строку
void Read_Adc()
   {
        ADCON0=0b11000001;              // Выбор Analog0 RC_Mode и ADON
        ADCON0.GO=1;                    // Запуск преобразования
        while(ADCON0.GO);               // Ожидаем, пока преобразование завершится
        Sensor0=(ADRESH*4)+(ADRESL/64); // 10 бит Данных ==> sensor0
        ADCON0=0b11000101;              // Выбор Analog1 RC_Mode и ADON
        ADCON0.GO=1;                    // Запуск преобразования
        while(ADCON0.GO);               // Ожидаем, пока преобразование завершится
        Sensor1=(ADRESH*4)+(ADRESL/64); // 10 бит Данных ==> sensor1
   }

void main()
{
   Delay_ms(1000);                      // Начальное ожидание
   Lcd_Init(&PORTD);                    // Инициализация ЖКИ
   ANSEL = 0xFF;                        // PORTA ==> Analog
   TRISA = 0xFF;                        // PORTA ==> input
   Lcd_Cmd(LCD_CURSOR_OFF);             // Курсор ЖКИ выключен

   while(1)
   {
        Read_Adc();
        WordToStr(Sensor0,Txt);         // Преобразование значения Sensor0 в строку
        Lcd_Out(1,1,Txt);               // и отображение ее на экране ЖКИ
        WordToStr(Sensor1,Txt);         // Преобразование значения Sensor1 в строку
        Lcd_Out(2,1,Txt);               // и отображение ее на экране ЖКИ
   }
}
```

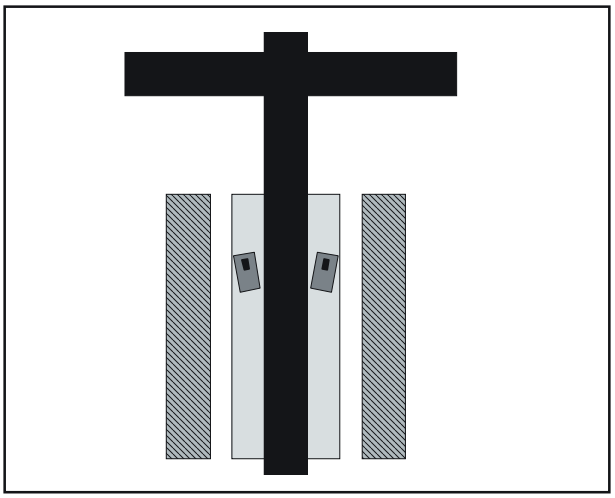
Распечатка A8-1: Код для чтения данных с рефлективного датчика ZX-03.

Если различие в коэффициенте отражения от белой и черной поверхностей невелико, то необходимо уменьшить расстояние от датчика до поверхностей. Если датчик расположен слишком далеко от поверхности, то прочитанное значение будет близко к нулю.

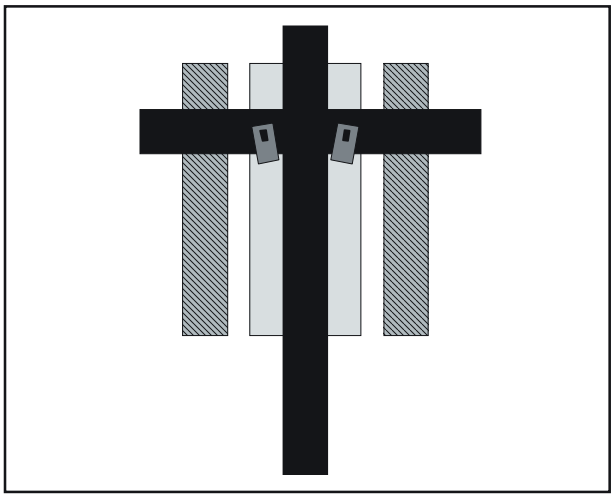
Задание 9

Движение вдоль черной линии

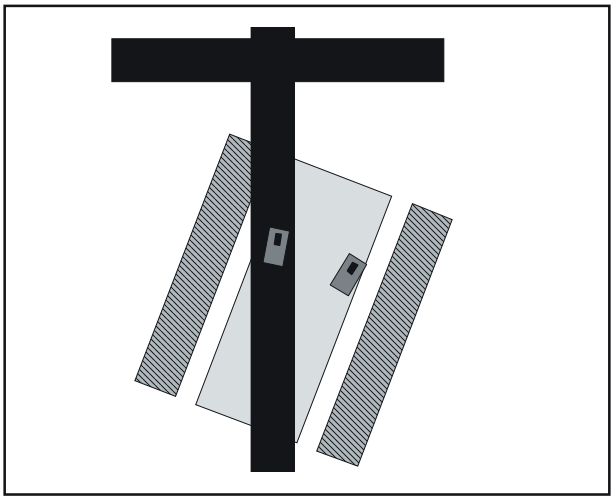
После установки Инфракрасных рефлекторов ZX-03 в Robo-PICA можно наблюдать 4 сценария его поведения при отслеживании линий, как показано на рисунке ниже:



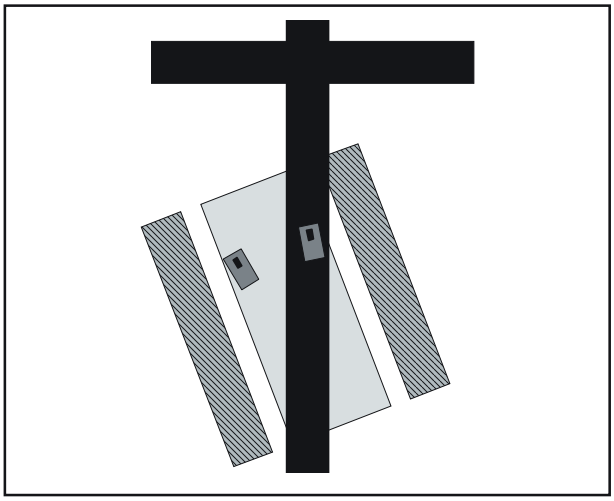
1. Оба датчика над белой поверхностью
Здесь показано, как робот движется вдоль черной линии



2. Оба датчика над черной поверхностью
Здесь показано, что два датчика расположены над поверхностью черной линии. Робот может выбрать дальнейшие действия из нескольких альтернативных вариантов: повернуть налево, направо, остановиться, поехать назад и т.п.



3. Левый датчик над черной поверхностью.
Это значит, что робот должен повернуть налево, чтобы вернуться в рабочее положение



4. Правый датчик над черной поверхностью.
Это значит, что робот должен повернуть направо, чтобы вернуться в рабочее положение

Из поведения при движении вдоль черной линии, можно составить алгоритм для написания программы управления Robo-PICA, которая показана на распечатке A9-1. Последовательность выполнения программой операций следующая:

1. Установить значения по умолчанию для взаимодействия с модулем аналого-цифрового преобразователя в микроконтроллере PIC16F887.
2. Разрешить обе схемы управления двигателями постоянного тока.
3. Получить данные от обоих сенсоров, которые подключены к выводам портов RA0 и RA1. Сохранить данные в переменной SENSOR0 (сохраняет данные от RA0) и SENSOR1 (сохраняет данные от RA1)
4. Сравнить полученные данные с пороговым значением. *Если значения SENSOR0 и SENSOR1 больше, чем пороговое значение, управлять роботом для движения вперед.*
5. *Если значения SENSOR0 и SENSOR1 меньше, чем пороговое значение, управлять роботом для движения вперед.* Это условие того, что робот встретился с пересечением линий и принимает решение продолжать двигаться вперед.
6. *Если только значение SENSOR0 меньше, чем пороговое значение, робот поворачивает влево.*
7. *Если только значение SENSOR1 меньше, чем пороговое значение, робот поворачивает вправо.*

A9.1 Наберите текст программы, показанный на распечатке A9-1. Скомпилируйте и загрузите код в Robo-PICA.

A9.2 Выключите питание и отсоедините от Robo-PICA загрузочный кабель.

A9.3 Поместите Robo-PICA на перекрестье черных линий на демонстрационном бумажном листе (входит в комплект робота Robo-PICA).

A9.4 Включите напряжение питания. Наблюдайте за функционированием робота.

Робот Robo-PICA будет передвигаться, следуя черной линии, никаким образом не реагируя на пересечение линий.

Как высчитать пороговое значение?

Если датчик читает значение большее, чем пороговое значение, то можно рассматривать обнаруженную поверхность как **"Черную"**

Если датчик читает значение большее, чем пороговое значение, то можно рассматривать обнаруженную поверхность как **"Белую"**

Пороговое значение можно вычислить следующим образом:

(Минимальное значение, считываемое сенсором с белой поверхности + Максимальное значение, считываемое сенсором с черной поверхности) / 2

Пример

Если минимальное значение, считываемое сенсором с белой поверхности равно 300 и максимальное значение, считываемое сенсором с черной поверхности равно 100,

Пороговое значение равно (300+100) / 2 = **200**

```
/*******************************************************/
/***** Движение вперед вдоль черной линии при обнаружении пресечения **********/
/*******************************************************/
#include <motor.h>
int Sensor0,Sensor1;                    // Хранятся Аналоговые значения
char Txt[6];                            // Хранятся значения, преобразованные в строки

void Read_Adc()
   {
      ADCON0=0b11000001;                // Выбрать Analog0 RC_Mode и ADON
      ADCON0.GO=1;                      // Запустить преобразование
      while(ADCON0.GO);                 // Ожидаем, пока преобразование закончится
      Sensor0=(ADRESH*4)+(ADRESL/64); // 10 бит Данных ==> sensor0
      ADCON0=0b11000101;                // Выбрать Analog1 RC_Mode и ADON
      ADCON0.GO=1;                      // Запустить преобразование
      while(ADCON0.GO);                 // Ожидаем, пока преобразование закончится
      Sensor1=(ADRESH*4)+(ADRESL/64); // 10 бит Данных ==> sensor1
   }

void main()
{
   Delay_ms(1000);                      // Начальное ожидание
   Lcd_Init(&PORTD);                    // Инициализировать ЖКИ
   ANSEL = 0xFF;                        // PORTA ==> Аналоговый
   TRISA = 0xFF;                        // PORTA ==> вход
   Lcd_Cmd(LCD_CURSOR_OFF);             // Погасить курсор ЖКИ

   while(1)
   {
      Read_Adc();
      WordToStr(Sensor0,Txt);           // Преобразовать Sensor0 в строку
      Lcd_Out(1,1,Txt);                 // и отобразить на ЖКИ
      WordToStr(Sensor1,Txt);           // Преобразовать Sensor1 в строку
      Lcd_Out(2,1,Txt);                 // и отобразить на ЖКИ

      if ((Sensor0>500)&(Sensor1>500))  // Если оба датчика над белым
      Forward(255);                     // Двигаться ВПЕРЕД
      if ((Sensor0<500)&(Sensor1<500))  // Если оба датчика над черным
      {
      Forward(255);                     // Вперед 0.3 секунды
      Delay_ms(300);
      }
      if (Sensor0<500)                  // Только Sensor0 обнаружил линию
      {
        S_Left(255);                    // поворот Налево
      }
      if (Sensor1<500)                  // Только Sensor1 обнаружил линию
      {
        S_Right(255);                   // Поворот Направо
      }
   }
}
```

Распечатка А9-1: Программа на языке С управления движением робота вдоль линии

Выводы

В качестве основы операции движения робота вдоль линии можно отметить 3 фактора. Во-первых, быстродействие датчиков. Во-вторых, место установки и взаимное расположение датчиков и, в-третьих, управляющая программа. В некоторых случаях роботу требуется большее количество датчиков для обнаружения сложных комбинаций линий, таких как 3 перекрещивающиеся линии, скрещивающиеся линии и т.п. См. Рисунок A9-1.

При установке инфракрасных рефлекторов для обнаружения линий, очень важным является расстояние между датчиками. Правильный выбор этого расстояния увеличивает чувствительность системы. Если датчики расположены слишком близко к линии, робот будет обнаруживать линию много раз. В таком случае робот будет совершать беспорядочные движения вправо и влево, напоминающие движение змея. См. Рисунок A9-2. При этом скорость движения может значительно упасть.

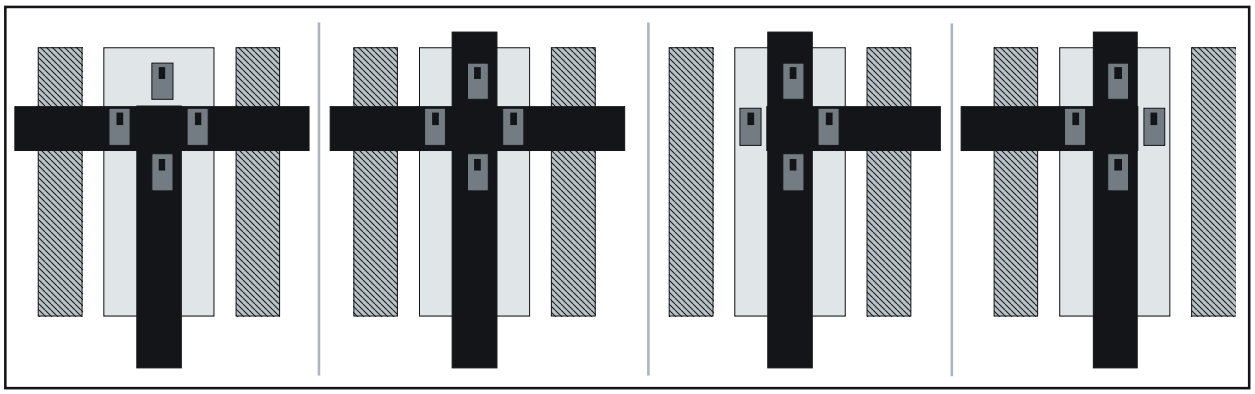


Рисунок A9-1: Добавление дополнительных датчиков для обнаружения сложных фигур

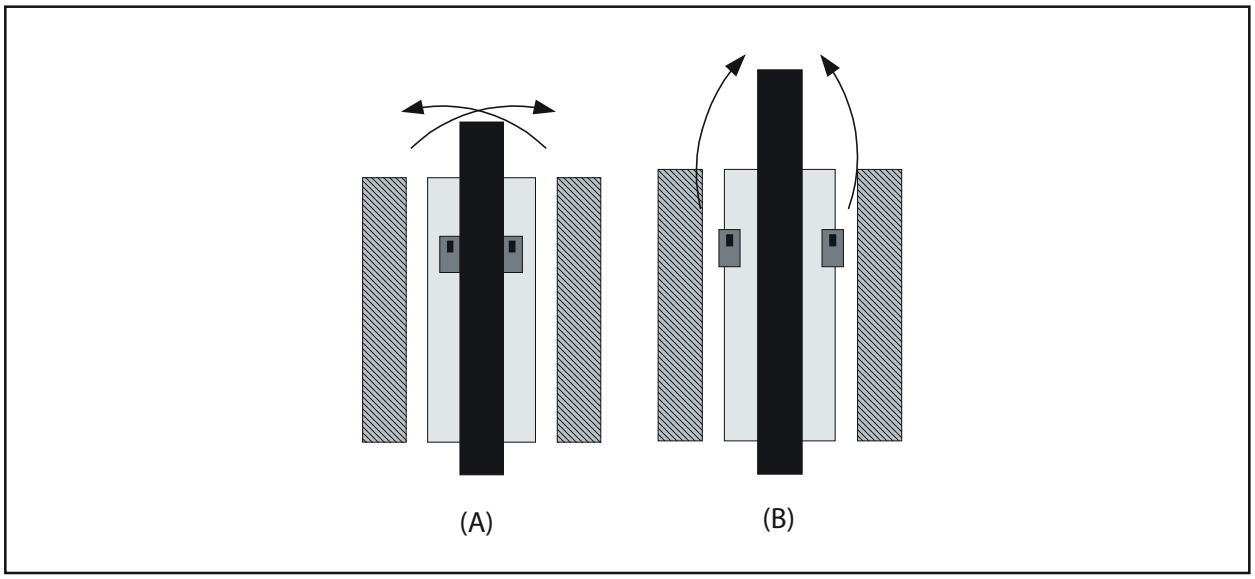


Рисунок A9-2: Результат установки датчиков линии
(A) Установка датчиков слишком близко друг к другу. Может привести к пляске робота.
(B) Установка датчиков далеко друг от друга и от линии.
Если расстояние выбрано достаточно хорошо, то это может помочь роботу двигаться лучше вдоль линии.

Глава 7

Робот с дистанционным управлением

Другой особенностью автоматических роботов является возможность приема команд с большого расстояния, используя инфракрасные лучи. Это напоминает робота с проводным дистанционным управлением, за исключением того, что команды принимаются через последовательный порт. Робот Robo-PICA имеет инфракрасный пульт дистанционного управления, называющийся ER-4. Пульт дистанционного управления ER-4 модулирует инфракрасное излучение сериями импульсов. Вначале в Robo-PICA необходимо установить модуль инфракрасного приемника на частоту 38 кГц для приема команд управления с пульта ER-4.

7.1 Модуль инфракрасного приемника ZX-IRM

Данные, отправленные с Инфракрасным Излучением, могут преодолевать расстояния от 5 до 10 метров, как это происходит при дистанционном управлении телевизорами. Несущая частота модуляции обычно выбирается 38 кГц. Приемник должен демодулировать несущую частоту 38 кГц. После этого он передает продетектированный сигнал как последовательность данных в микроконтроллер. Часто, для повышения чувствительности приемника, демодулятор выполняется по схеме синхронного детектора.

Если сенсор не обнаруживает в инфракрасном излучении частоту 38 кГц, выходной сигнал принимает значение логической "1". Иначе, если частота 38 кГц обнаружена, выходной сигнал принимает значение логического "0".

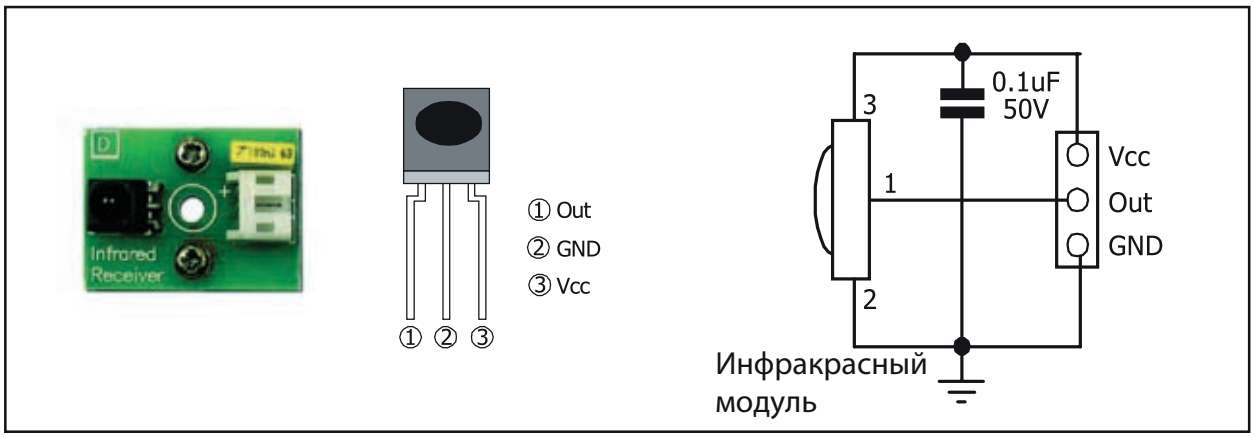


Рисунок 7-1: Показана фотография модуля Инфракрасного Приемника, назначение выводов датчика и схема его включения

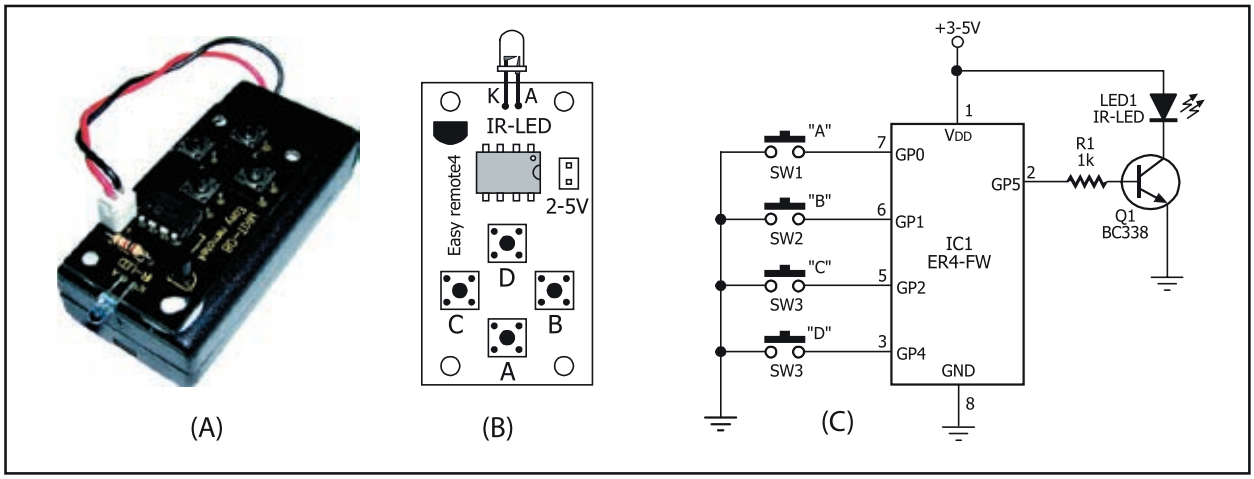


Рисунок A7-2: Показывает фотографию, расположение элементов на печатной плате и принципиальную схему простейшего пульта дистанционного управления ER-4.

7.2 Инфракрасный Пульт Дистанционного управления ER-4

- Работает на расстояниях от 4 до 8 метров на любом открытом пространстве.
- Четырехканальный выключатель работает в режиме включено/выключено
- Потребляет мало энергии; после отправки команды автоматически переходит в режим пониженного энергопотребления.
- Работает при напряжении питания 2,4...3 В от двух батарей AA – как обычных, так и перезаряжаемых.
- Передает последовательность данных, используя стандарт RS-232 при скорости передачи 1200 бод и формате данных 8N1 (8 бит данных, без четности, 1 стоповый бит)

7.2.1 Формат данных, передаваемых пультом ER-4

Чтобы сделать удобным для приемника чтение информации о нажатой на пульте дистанционного управления кнопке, пульт ER-4 передает последовательность данных согласно стандарту RS-232, при скорости передачи 1200 бод и формате данных 8N1. Символы передаются в соответствии с нажатой на пульте дистанционного управления кнопкой. Расположение кнопок показано на рисунке A7-2:

Нажатие кнопки A, большая A , сопровождается передачей маленькой A (a)

Нажатие кнопки B, большая B, сопровождается передачей маленькой B (b)

Нажатие кнопки C, большая C, сопровождается передачей маленькой C (c)

Нажатие кнопки D, большая D, сопровождается передачей маленькой D (d)

Основанием для наличия альтернативных больших и маленьких букв является то, что приемник может различать, будет ли кнопка удерживаться непрерывно, или пользователь будет совершать повторные нажатия. Если пользователь совершает повторные нажатия, то вначале будет послана большая буква. Если пользователь повторно нажмет на одну и ту же кнопку еще раз, вторым символом будет послана маленькая буква. Если пользователь будет удерживать кнопку непрерывно, последняя буква будет передаваться непрерывно.

Задание 10

Чтение команд дистанционного управления

На Распечатке А10-1 представлена программа на языке С для чтения данных с пульта дистанционного управления ER-4 и отображения их на экране ЖКИ. Кроме того, эти данные используются для сравнения с образцовыми данными для управления звуком пьезоизлучателя, генерируемым функцией Sound_Play компилятора mikroC.

А10.1 Наберите текст программы, показанный на распечатке А10-1. Скомпилируйте и загрузите код в Robo-PICA.

```
char *text = "ER-4 Remote";                    // Определение текста сообщения
unsigned char ir_cmd=0;                        // Сохраняет символ команды от ПДУ ER-4
//-------- Подпрограмма обработки прерывания INT ----------------//
void interrupt()
{
   unsigned char i;                            // Сохраняет значение счетчика
   if(INTCON.INTF)                             // Проверяет флаг прерывания RB0 (по спаду импульса)
   {
        Delay_us(416);                         // Задержка на 1/2 или 1 периода импульса
                                               // (скорость передачи 1200 бод)
        for(i=0;i<8;i++)                       // Цикл из 8 шагов для сохранения данных от ER-4
        {
             Delay_us(833);                    // Задержка на 1 период импульса
                                               // (скорость передачи 1200 бод)
             ir_cmd = ir_cmd>>1;               // Сдвиг на 1 бит вправо
             if((PORTB & 0x01)==1)             // Выполнить логическое И RB0 = '1'?
               ir_cmd = ir_cmd | 0x80;         // Вставить бит, если результат '1'
        }
        Delay_us(833);                         // Задержка на 1 период импульса
                                               // (скорость передачи 1200 бод)
        INTCON.INTF =0; // Очистить флаг прерывания
   }
}
//----------------- Функция для получения символа от пульта ---------//
unsigned char get_remote()
{
   unsigned char _key=ir_cmd;                  // Поместить символ в буфер
   ir_cmd=0;                                                  // Очистить старое значение
   return(_key);                                          // Вернуть символ от ПДУ
}
//----------------- Главная программа ---------------------------------//
void main()
{
   unsigned char key;                          // Save Remote Key Press
   ANSELH.F4=0;                                // RB0 ==> Digital IO
```

Распечатка А10-1: Программа чтения данных от пульта дистанционного управления ER-4 (начало)

```
    OPTION_REG.INTEDG = 0;                  // INT falling edge
    INTCON.INTE =1;                         // Enable INT/PB0
    INTCON.GIE =1;                          // Enable Global interrupt
    Lcd_Init(&PORTD);                       // Инициализация ЖКИ, подключенного к PORTD
    Lcd_Cmd(Lcd_CLEAR); // Очистить дисплей
    Lcd_Cmd(Lcd_CURSOR_OFF); // Погасить курсор
    Lcd_Out(1, 1, text); // Вывести текст на ЖКИ,2-я строка,1-1 столбец
    Sound_Init(&PORTC,0);
    while(1) // Infinite loop
    {
        key = get_remote(); // Принять символ от ПДУ
        if(key=='a' || key=='A') // Нажата кнопка A?
        {
            Lcd_Out(2, 1, "Button A Press "); // Вывести сообщение
            Sound_Play(100,500);
        }
        else if(key=='b' || key=='B') // Нажата кнопка B?
        {
            Lcd_Out(2, 1, "Button B Press "); // Вывести сообщение
            Sound_Play(110,500);
        }
        else if(key=='c' || key=='C') // Нажата кнопка C?
        {
            Lcd_Out(2, 1, "Button C Press "); // Вывести сообщение
            Sound_Play(120,500);
        }
        else if(key=='d' || key=='D') // Нажата кнопка D?
        {
            Lcd_Out(2, 1, "Button D Press "); // Вывести сообщение
            Sound_Play(130,500);
        }
    }
}
```

Распечатка A10-1: Программа чтения данных от пульта дистанционного управления ER-4 (окончание)

A10.2 Вставить две батареи размера AA в батарейный отсек пульта ДУ ER-4.

A10.3 Включить питание робота. Нажать кнопку на пульте ДУ ER-4, чтобы отправить данные в приемник ZXIRM робота Robo-PICA. Наблюдать результат операции на ЖКИ-индикаторе и пьезоизлучателе.

ЖКИ будет показывать буквы A, B, C, D или a,b,c,d, следуя за нажатием кнопок на пульте Дистанционного Управления ER-4, и при этом будет раздаваться звук различной частоты.

Задание 11

Управление движением Robo-PICA на ИК-лучах

В этом задании показано как управлять движением Robo-PICA посредством Инфракрасного пульта дистанционного управления (ПДУ) ER-4. При нажатии на кнопки пульта дистанционного управления ER-4 робот будет двигаться в разных направлениях: вперед, назад, вправо и влево.

A11.1 Введите текст программы, показанный на Распечатке A11-1. Скомпилируйте и загрузите код в Robo-PICA.

A11.2 Выключите питание и отсоедините загрузочный кабель от Robo-PICA.

A11.3 Положите Robo-PICA на пол.

A11.4 Включите питание. Используйте пульт дистанционного управления ER-4 и наблюдайте за работой робота.

Робот Robo-PICA не двигается, пока не нажата ни одна из кнопок ER-4. Нажимая на кнопки пульта дистанционного управления ER-4, следите за тем, чтобы приемный датчик находился в прямой видимости, при этом связь будет наиболее надежной.

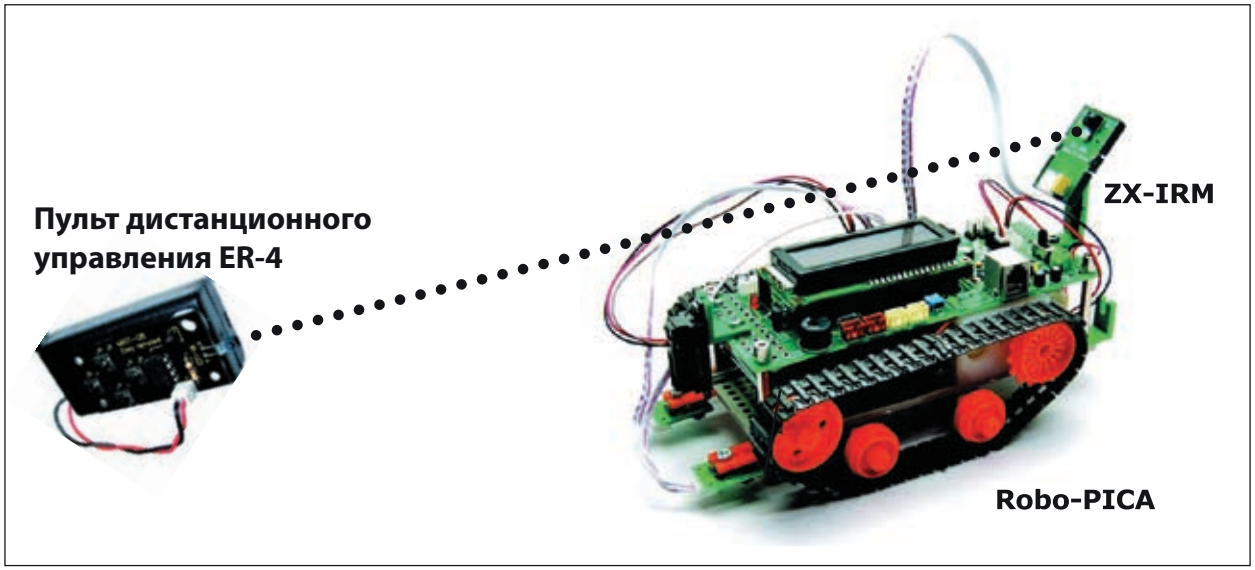


Рисунок A11-1: Показывает управление *Robo-PICA при помощи* пульта дистанционного управления *ER-4*.

```
#include <motor.h>
char *text = "ER-4 Remote";                    // Определить сообщение
unsigned char ir_cmd=0;                        // Получит значение нажатой кнопки для ПДУ ER-4
//------------------ Сервисная подпрограмма прерывания INT -----------------//
void interrupt()
{
   unsigned char i;                            // Хранит значение счетчика
   if(INTCON.INTF)                             // Проверяет значение флага прерывания RB0 (спад импульса)
   {
        Delay_us(416);                         // Задержка на время 1/2 или 1 бита(скорость 1200 бод)
        for(i=0;i<8;i++)                       // Цикл на 8 шагов для сохранения данных от ПДУ ER-4
        {
             Delay_us(833); /                  / Задержка на время 1 бита(скорость 1200 бод)
             ir_cmd = ir_cmd>>1;               // Сдвиг на 1 бит вправо
             if((PORTB & 0x01)==1)             // Выполнить логическое И. RB0 = '1'?
              ir_cmd = ir_cmd | 0x80;          // Вставить бит, если '1'
        }
        Delay_us(833);                         // Задержка на время 1 бита(скорость 1200 бод)
        INTCON.INTF =0;                        // Очистка флага прерывания
   }
}

//------------------ Функция получения символа от ПДУ ------------------//
unsigned char get_remote()
{
   unsigned char _key=ir_cmd;                  // Записать символ в буфер
   ir_cmd=0;                                   // Стереть старые данные
   return(_key);                               // Вернуть символ от ПДУ
}

//------------------ Основная Программа ---------------------------------//
void main()
{
   unsigned char key;                          // Хранит значение нажатой кнопки ПДУ
   ANSELH.F4=0;                                // RB0 ==> Цифровой Ввод/Вывод
   OPTION_REG.INTEDG = 0;                      // Полярность сигнала прерывания
   INTCON.INTE =1;                             // Разрешить INT/PB0
   INTCON.GIE =1;                              // Разрешить глобальные прерывания

   Lcd_Init(&PORTD);                           // Инициализировать ЖКИ, присоединенный к порту PORTD
   Lcd_Cmd(Lcd_CLEAR);                         // Очистить дисплей
   Lcd_Cmd(Lcd_CURSOR_OFF);                    // Запретить курсор ЖКИ
   Lcd_Out(1, 1, text);                        // Вывести текст на ЖКИ, 2-я строка, 1-й столбец
   Sound_Init(&PORTC,0);
   while(1)                                    // Бесконечный цикл
{
        if (ir_cmd==0)
        {
             Motor_Stop();
        }
        else
        {
             key = get_remote();               // Прочитать данные с датчика ДУ
```

Распечатка A11-1: Программа на языке C для Robo-PICA, иллюстрирующая использование модуля удаленного управления (начало)

```
            if(key=='a' || key=='A')              // Нажата кнопка A?
        {
            Lcd_Out(2, 1, "Button A Press");  // Показать сообщение
                                                  // Нажата кнопка A
            Backward(255);Delay_ms(50);
        }
        else if(key=='b' || key=='B')             // Нажата кнопка B?
        {
            Lcd_Out(2, 1, "Button B Press");  // Показать сообщение
                                                  // Нажата кнопка B
            S_Right(255);Delay_ms(50);
        }
        else if(key=='c' || key=='C')             // Нажата кнопка C?
        {
            Lcd_Out(2, 1, "Button C Press");  // Показать сообщение
                                                  // Нажата кнопка C
            S_Left(255);Delay_ms(50);
        }
        else if(key=='d' || key=='D')             // Нажата кнопка D?
        {
            Lcd_Out(2, 1, "Button D Press");  // Показать сообщение
                                                  // Нажата кнопка D
            Forward(255);Delay_ms(50);
            }
        }
    }
}
```

Распечатка A11-1: Программа на языке C для Robo-PICA, иллюстрирующая использование модуля удаленного управления (окончание)

Приложение A

Активация Лицензионного ключа компилятора mikroC

О Свободной Версии

Последняя версия компилятора всегда доступна для загрузки с веб-сайта mikroE. Это полнофункциональное программное обеспечение со всеми библиотеками, примерами, и разнообразными разделами помощи. Кроме того, отдельно можно загрузить руководство в PDF-формате для последующей его распечатки.

Свободно загруженная версия имеет только одно ограничение: размер выходного HEX-файла не может превышать 2к программных слов. Несмотря на то, что это может казаться ограничением, такой вариант компилятора позволяет создавать практические, работающие приложения, даже не вспоминая об ограничении демо-версии. Если в компиляторе необходимо создавать реально сложные проекты, следует рассмотреть вариант покупки лицензионного ключа на полную версию компилятора.

Замечание: Лицензионный ключ остается действительным до переформатирования жесткого диска. В случае, если жесткий диск был отформатирован, следует запросить новый ключ активации. При переустановке операционной системы Windows без форматирования жесткого диска лицензионный ключ все еще остается действительным.

Поддержка

Если возникли любые вопросы по поводу активации компилятора, посетите форум поддержки www.mikroe.com/forum/. Если возникли серьезные проблемы при использовании любого из продуктов mikroE или необходимо получить дополнительную информацию, свяжитесь, пожалуйста, непосредственно с mikroE.

Контактная информация mikroE:

mikroElektronika
Телефоны: + 381 (11) 30 66 377, + 381 (11) 30 66 378
Факс: + 381 (11) 30 66 379
Web: www.mikroe.com
E-mail: office@mikroe.com

Авторские права

Особая благодарность компании **mikroElektronika** (**www.mikroe.com**) за демо-версию программного обеспечения mikroC и компании **Microchip Technology**, производителю микроконтроллера PIC16F84, за поддержку программного обеспечения PICkit2™.

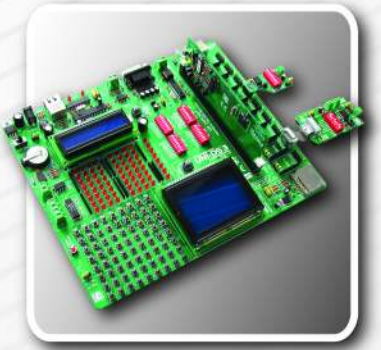
ТЕРРА
ЭЛЕКТРОНИКА
www.terraelectronica.ru

mikroElektronika
DEVELOPMENT TOOLS | COMPILERS | BOOKS

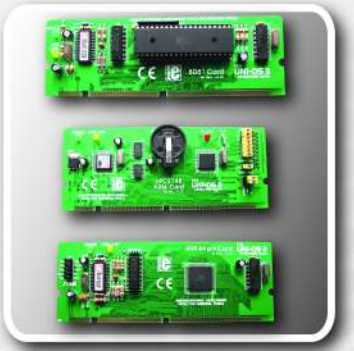
СРЕДСТВА РАЗРАБОТКИ И ОТЛАДКИ

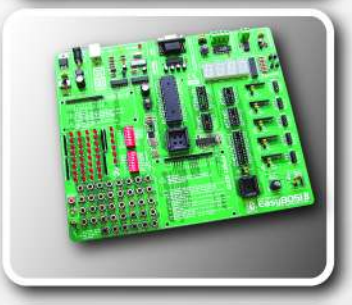
для 8-, 16-, 32-разрядных микроконтроллеров и «систем на кристалле»

АППАРАТНЫЕ СРЕДСТВА РАЗРАБОТКИ

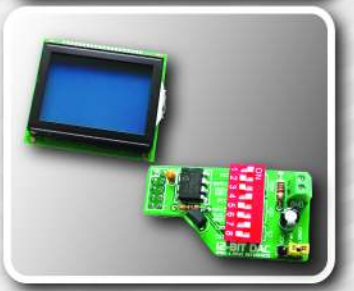


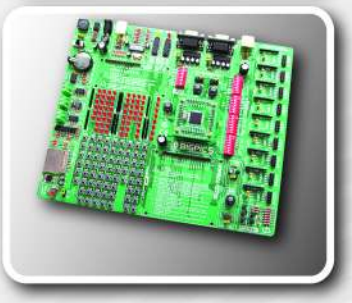
Универсальный стенд UNI-DS3 и микроконтроллерные платы специализации (8051, AVR, PIC, dsPIC, ARM, PSoC)



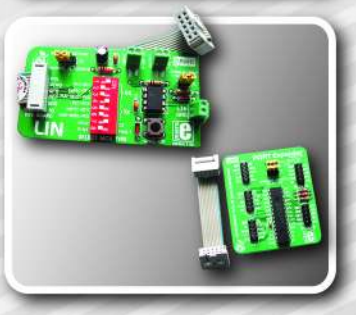


Стенды Easy8051, EasyARM, EasyAVR5, EasyPIC5, EasydsPIC4, EasyPSOC3, EasyHC-908 с необходимыми модулями ввода/вывода





Стенды BIGAVR, BIGPIC5, BIGdsPIC, dsPICPRO4 с расширенным набором модулей ввода/вывода



ПРОГРАММНЫЕ СРЕДСТВА РАЗРАБОТКИ

Компиляторы

- mikroC
- mikroPascal
- mikroBasic

ТЕРРАЭЛЕКТРОНИКА – официальный дистрибьютор компании mikroElektronika

ТЕРРА
ЭЛЕКТРОНИКА
www.terraelectronica.ru

НОВЫЕ ЭЛЕКТРОННЫЕ
КОМПОНЕНТЫ ДЛЯ РАЗРАБОТЧИКОВ

- Интегральные микросхемы
- Оптоэлектронные приборы
- Дискретные приборы и силовые модули
- Датчики и преобразователи физических величин
- Системы отображения видеоинформации
- Источники электропитания модульные
- Компоненты для устройств радиоидентификации
- Компоненты для беспроводных систем передачи данных

Новинки электронных компонентов от мировых хорошо известных…

…и перспективных молодых компаний

Ваш дилер: