Содержание

Вступление

На сегодняшний день, когда механические системы разного уровня сложности внедряются как в бытовую, так и в промышленную технику, а выполняемые ими функции становятся всё сложнее и разнообразнее, особенно остро стоит вопрос управления ими без непосредственного контроля человека. Для этого используются автоматические системы на основе ПЛИС. Однако поскольку производство таких систем отдельно друг от друга невозможно, возникает необходимость проектирования таких систем с уже встроенными элементами электроники в механической системе. Этими вопросами занимается такая наука, как мехатроника.

Мехатроника - это область науки и техники, основанная на синергетическом объединении узлов точной механики с электронными, электротехническими и компьютерными компонентами, обеспечивающими проектирование и производство качественно новых модулей, систем, машин и систем с интеллектуальным управлением их функциональными движениями.

Мехатроника сосредоточена на механике, электронике, вычислении, технике автоматического управления, молекулярных разработок (из нанохимии и биологии), оптических разработок, которые, объединившись, делают возможным генерацию более простых, более экономичных, надежных и универсальных систем (см. рис. 1.1).

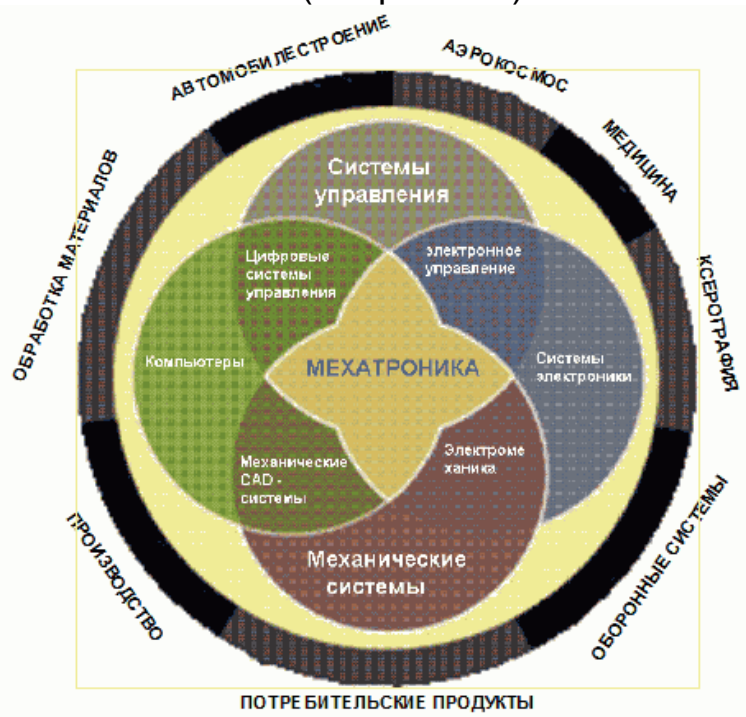


Рисунок 1.1 – Структурная схема взаимосвязи мехатроники с другими науками

Обычно мехатронику представляют как единство 3-х частей (рис. 1.2) – привода (1), исполнительных и передаточных устройств (2) и управления (3). Область 4 традиционно называют электромеханикой, 5 – автоматикой, 6 – областью регулируемого привода, а 7 – ядром направления мехатроники.

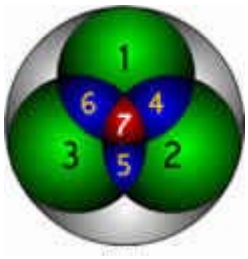


Рисунок 1.2 – Мехатроника – единство трёх частей

Слово "мехатроника”, в собственном современном значении, впервые возникло в 1972 году в Стране восходящего солнца - Японии, при этом возникло на машиностроительном заводе, одновременно занимавшемся выпуском электронных устройств (см. рис. 1.3). Еще раньше, в 50-х годах возникло слово "мехатроны”, относившееся к электровакуумным приборам. Сейчас мы квалифицируем эти приборы как составляющие электромеханики. В общем возникновение мехатроники от электромеханикиочевидно.

Рисунок 1.3 – Машиностроительный завод

Техническая кибернетика имеет дело с вопросом техники автоматического управления мехатронных систем и используется, чтобы управлять или регулировать такую систему. Через взаимодействие

мехатронные модули выполняют цели производства и наследуют гибкие и быстрые производственные свойства в промышленной схеме. Современное промышленное оборудование состоит из мехатронных модулей, которые объединены согласно архитектуре управления.

Наиболее известная архитектура включает иерархию:

а) коллегиальное управление;

б) организованная система, изобилующая перекрытием, множественностью, смешанным доминированием, и/или расходящимися, но сосуществующими моделями отношений;

в) гибридная, где методы достижения технического эффекта описаны алгоритмами управления, которые могли бы или не могли использовать формальные методы в проекте управления.

Гибридные системы, что важно для мехатроников, включают промышленные системы, накопители синергии, вездеходы для исследования планет, автомобильные подсистемы, такие, как антиблокировочные тормозные системы и системы вращения, помогают в каждодневной работе оборудования, таких как камеры с автофокусом, видео, жесткие диски компьютеров, CD-плэйеры.

Описание учебного лабораторного стенда IE-ROBO-51

IE-ROBO-51 (см. рис. 1.4) – набор разработчика на базе MCS51-микроконтроллера позволяет демонстрировать одновременно несколько базовых алгоритмов управления механической платформой на основе показаний группы датчиков, запрограммированной траектории и скорости каждого исполнительного органа, а также по командам пользователя. Для начального использования достаточно только собрать шасси с приводом (достаточно одной отвертки), установить необходимые датчики и запрограммировать МК. Все демонстрационные программы приведены на компакт-диске и доступны для адаптации к условиям конкретного приложения даже школьнику, т.к. разработаны они на бейсике. Это идеальный инструмент для изучения микроконтроллеров.

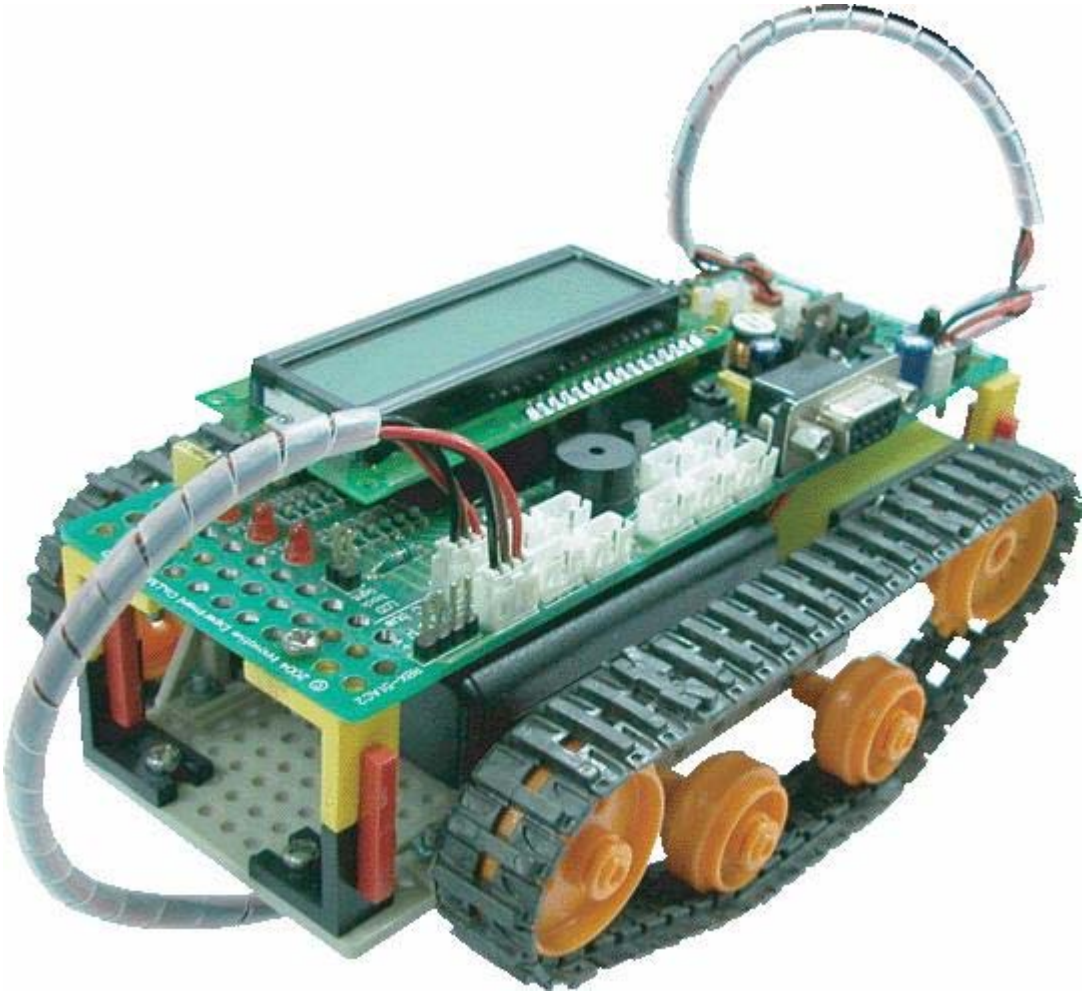


Рисунок 1.4 – Лабораторный стенд IE-ROBO-51

Отличительные особенности:

- плата на базе T89C51AC2;
- ЖКИ 16x2;
- два драйвера DC моторов;
- два электродвигателя с редукторами 1:48;
- три порта для сервомоторов;
- универсальное свободнопереконфигурируемое шасси;
- две пары инфракрасных датчиков;
- две платы с кнопками;
- инфракрасный приемник и пульт ДУ;
- контактные датчики;
- выводы МК доступны на разъемах;
- 3 светодиода;
- интерфейс RS-232 для программирования;
- среда программирования с поддержкой ассемблера, бейсика и Си;
- исходные коды программ с демонстрацией базовых функций компонентов.

Комплектация:

- отладочная плата с C51 микроконтроллером;
- кабель RS-232;
- ИК пульт управления **IE-ER-4**;
- ИК приемник **IE-ZX-IRM**;
- инфракрасный дальномер **GP2D120**;
- ЖКИ 16x2;
- два электродвигателя с редукторами **IE-BO2-48M**;
- платформа для создания самоходного робота **IE-UNIVERSAL PLATE SET**;
- шасси **IE-TRACKWHEEL KIT;**
- диск с программным обеспечением, документацией и исходными кодами;
- периферийные платы с разъёмами стандарта INEX: **IE-ZX-SWITCH** 2шт., **IE-ZX-03** 2шт.;
- подробное руководство по монтажу и программированию;
- крепёж и отвертка.

Области применения: приложения мехатроники; системы управления промышленным оборудованием; образовательные лаборатории; ознакомление с основами электроники и электротехники; изучение базовых принципов программирования электронных устройств.

Описание основных деталей лабораторного стенда IE-ROBO-51

Основные детали лабораторного стенда показаны на рис. 1.5

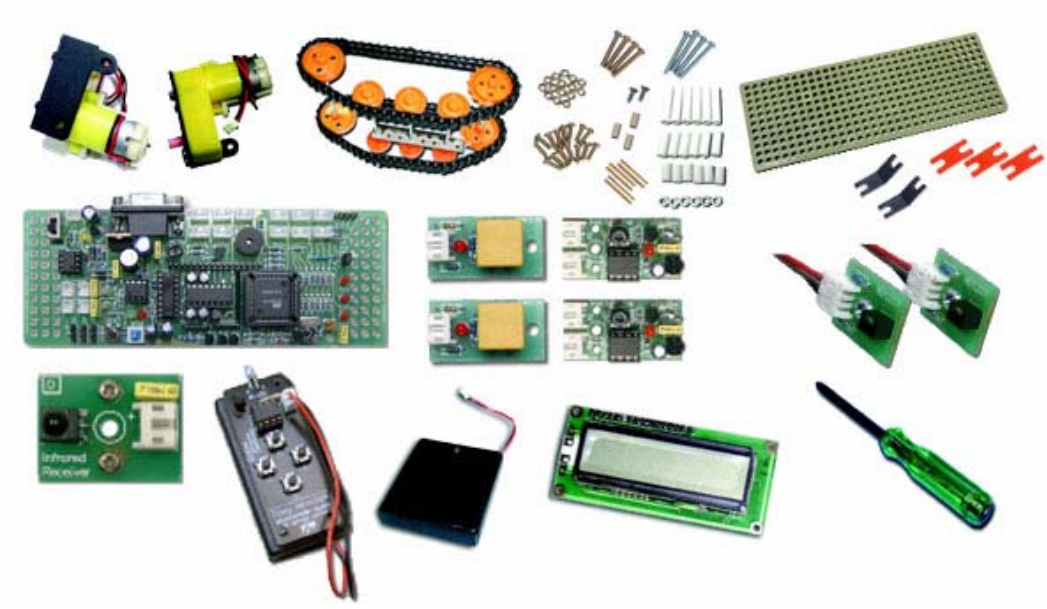


Рисунок 1.5 – Основные составляющие лабораторного стенда IE-ROBO-51

Механические комплектующие

Блок двигателя с редуктором (см. рис. 1.6) – состоит из корпуса с элементами крепления; двигателя постоянного тока, с напряжением питания 4,5 В (максимум 9 В) и потребляемым током 180 мА; редукторы с передаточным отношением 48:1 и максимальным моментом 4 кг/см.

Рисунок 1.6 – Блок двигателя с редуктором

Набор гусениц (см. рис. 1.7) состоит из гусениц трех типоразмеров, совместимых с множеством типов колес и гусеничных лент, осей и оснований.

Рисунок 1.7 – Набор гусениц

Монтажная плата и 4 вида различных цветных пластиковых крепежных пластин отображены на рис. 1.8 (10 плоских пластин, 10 прямоугольных пластин, 10 тупоугольных пластин и плоские пластины с 3/5/12 отверстиями).

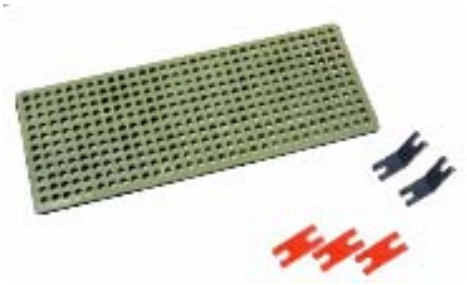


Рисунок 1.8 – Монтажная плата и крепежные пластины

Винты нескольких размеров и гайки (винты: 3х6 мм,3х10 мм, 3х15 мм, 3х25 мм и 3х35 мм, 3 мм гайки), винты с плоскими и закругленными головками, набор пластиковых втулок (длина: 3 мм, 15 мм и 25 мм), шестигранные стойки: 3х30 мм (см. рис. 1.9).

Рисунок 1.9 – Крепежные элементы

Электронные комплектующие

ZX-01 - концевые выключатели, показаны на рис. 1.10 (2 шт.).

Рисунок 1.10 – Концевые выключатели

GP2D120 – инфракрасный измеритель расстояния с дальностью 4-30 см (см. рис. 1.11).

Рисунок 1.11 – Инфракрасный измеритель расстояния

ZX-03 – инфракрасный отражатель 2 шт. (см. рис. 1.12).

Рисунок 1.12 – Инфракрасный отражатель

ZX-IR - инфракрасный приемник на частоту 38 кГц (см. рис. 1.13).

Рисунок 1.13 – Инфракрасный приемник

ER-4 - инфракрасный пульт дистанционного управления (см. рис. 1.14).

Рисунок 1.14 – Инфракрасный пульт дистанционного управления

Плата USB- программатора с ICD2 кабелем (см. рис. 1.15).

Рисунок 1.15 – Плата USB

USB- кабель (см. рис. 1.16).

Рисунок 1.16 – USB – кабель

ZX-POTH – потенциометр (см. рис. 1.17).

Рисунок 1.17 – Потенциометр

4 батареи AA (см. рис. 1.18).

Рисунок 1.18 – Батареи

RBX-877V2.0 - T89C51AC2 плата для робототехнических экспериментов (см. рис. 1.19).

Рисунок 1.19 – Плата для робототехнических экспериментов

Инструменты для сборки лабораторного стенда

Кусачки (см. рис. 1.20).

Рисунок 1.20 – Кусачки

Нож для бумаги со сменными лезвиями (см. рис. 1.21).

Рисунок 1.21 – Нож для бумаги со сменными лезвиями

Отвертка Филипс (см. рис. 1.22).

Рисунок 1.22 – Отвертка Филипс

Компьютер с установленной ОС Windows 98 или выше, имеющий как последовательный (RS232), так и параллельный порт (см. рис. 1.23).

Рисунок 1.23 – Компьютер

Лабораторная работа №1

Сборка мехатронной системы на базе IE-ROBO-51

Цель работы:

Изучить основные составляющие части мехатронной системы IE-ROBO-51, осуществить сборку мехатронной системы на базе IE-ROBO-51.

Основные теоретические сведения:

Прикрепите 2 коробки передач с двигателями постоянного тока к основанию. Поверните опресованную сторону правой коробки передач наружу, как показано на рис. 1.24.

Привинтите коробку передач винтами 3x10 мм с верхней стороны к основанию так, чтобы осталось свободное пространство возле левой коробки передач. Не затягивайте винты слишком сильно (см. рис. 1.25 - A2.2).

Оденьте ведущее ходовое колесо на ось коробки передач и зафиксируйте его саморезом 2 мм. Проделайте эту операцию для обеих коробок передач (см. рис. 1.25 - A2.2).

Поверните основание нижней стороной вверх (см. рис. 1.25 - A2.3). Прижмите длинную уголковую планку к основанию в определенном положении (см. рис. 1.27 - A2-6). Привинтите винтами 3x10 мм, предотвращая смещение планки относительно отверстий в панели.

Завинтите еще один винт 3x10 мм и гайку 3 мм в другое отверстие длинной уголковой планки (см. рис. 1.27 - A2-6).

Привинтите вторую длинную уголковую планку к основанию (см. рис. 1.26 - A2-4), вставив с верхней стороны винты 3x10 мм и затянув гайками 3 мм с нижней стороны (см. рис. 1.28 - A2-7).

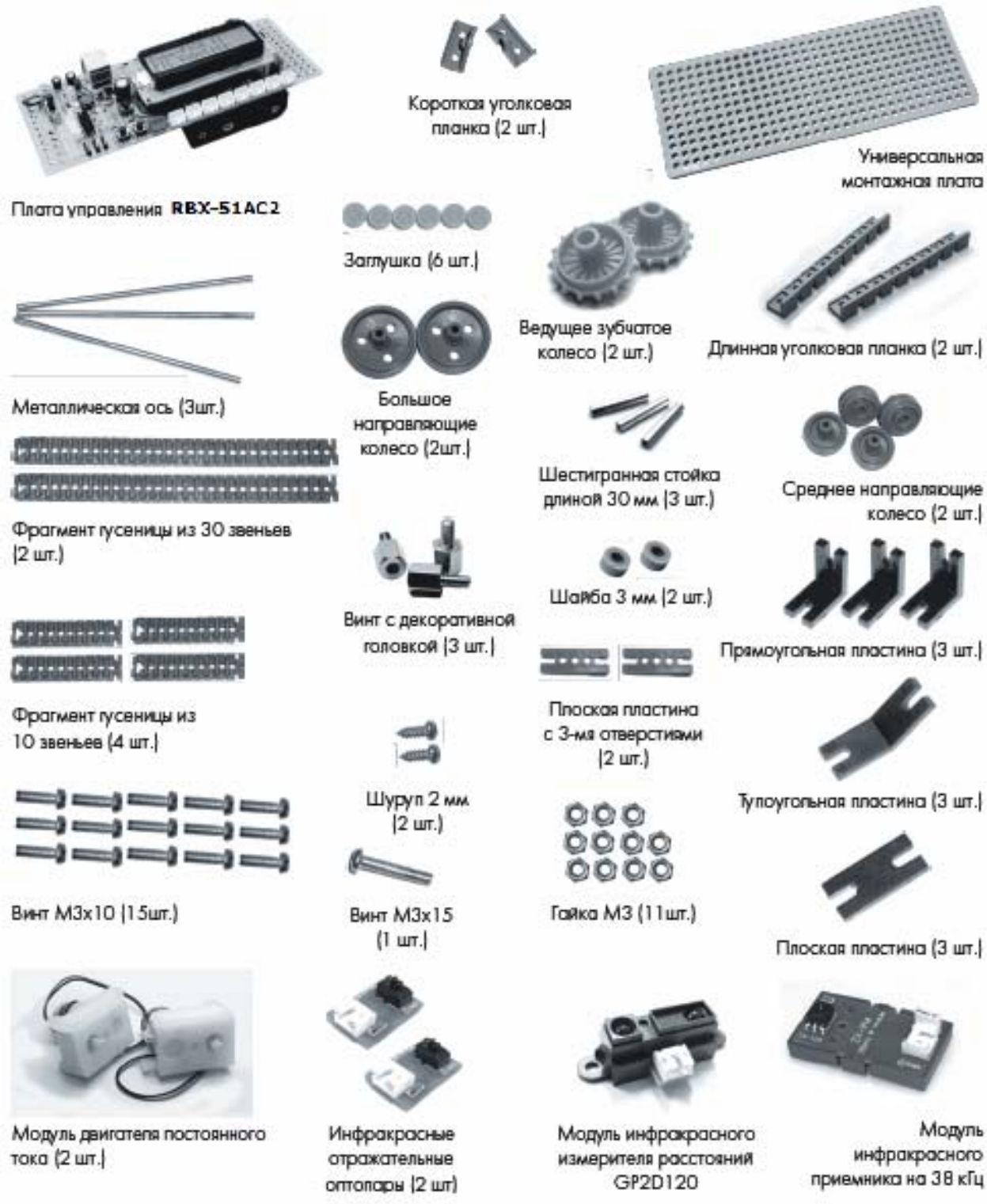


Рисунок 1.24 – Детали для сборки

Рисунок 1.25 – A2.2 и A2.3

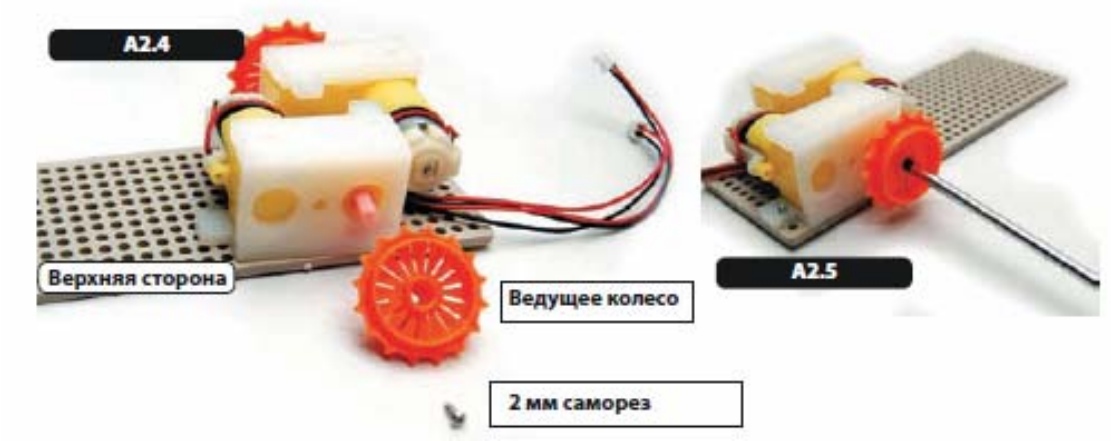


Рисунок 1.26 – A2.4 и A2.5

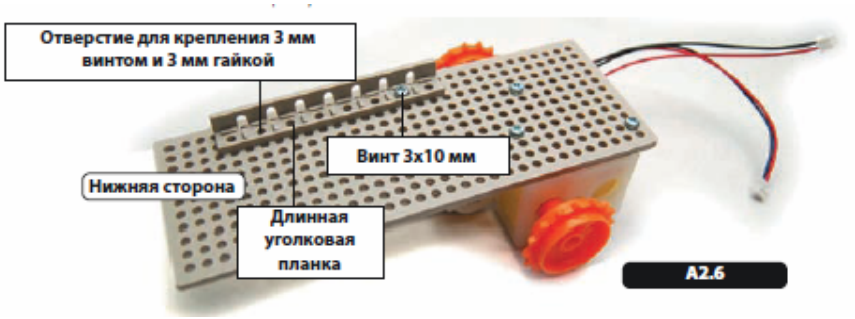


Рисунок 1.27 – A2.6

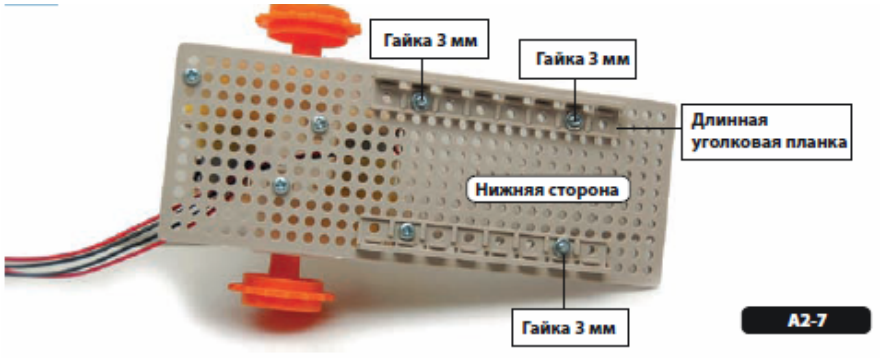


Рисунок 1.28 – A2.7

Поверните основание в обратную сторону (см. рис. 1.26 - A2.5). Присоедините 2 коротких уголковых планки к задней стороне основания робота (см. рис. 1.29 - A2.8), 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.

Привинтите шестигранную стойку с верхней стороны основания, повернув верхнюю сторону вниз и пропустив винт 3x10 мм через левое угловое отверстие и прямоугольную пластину (см. рис. 1.27 - A2.6).

Рисунок 1.29 – A2.8

Спереди присоедините 2 шестигранные стойки (см. рис. 1.28 - A2.7). 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.

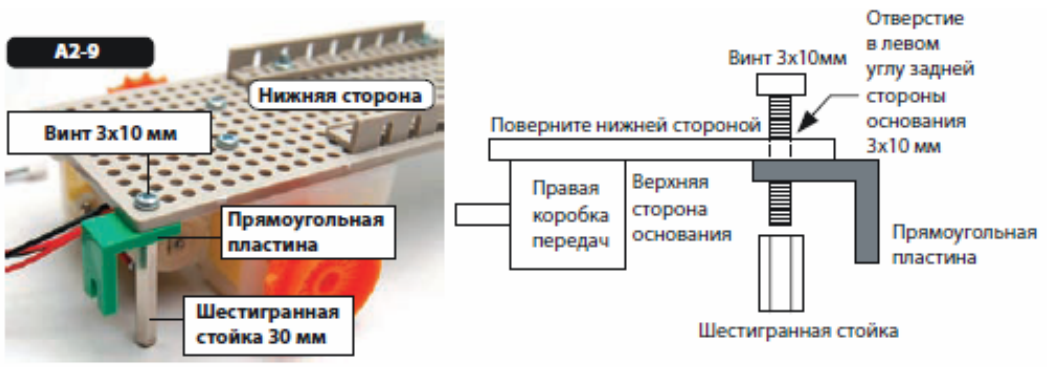


Рисунок 1.30 – A2.9

Оставьте конструкцию повернутой верхней стороной вниз (см. рис. 1.29 - A2.8). Вставьте металлические оси во второе и шестое отверстия длинной уголковой планки (см. рис. 1.32 - A2.11). Наденьте средние направляющие колеса на металлические оси. Вставьте в колеса заглушки таким образом, чтобы колеса плотно сидели на осях. Переверните основание.

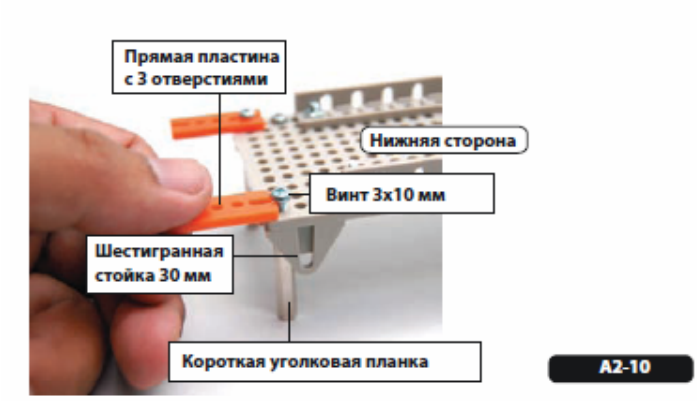


Рисунок 1.31 – A2.10

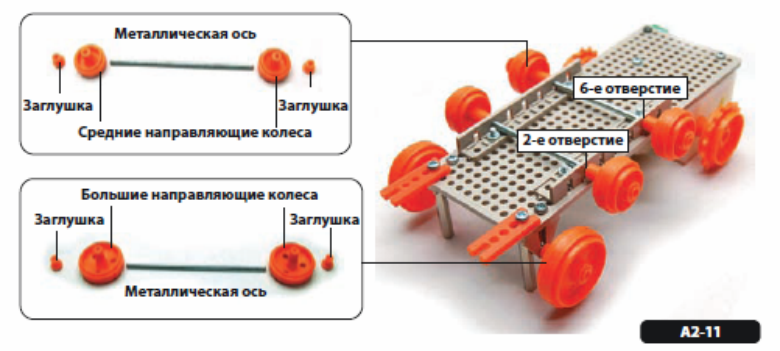


Рисунок 1.32 – A2.11

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.

Соберите две гусеницы, соединив вместе траки разного размера (см. рис. 1.30 - А2.9). Для изготовления одной гусеницы необходим 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.

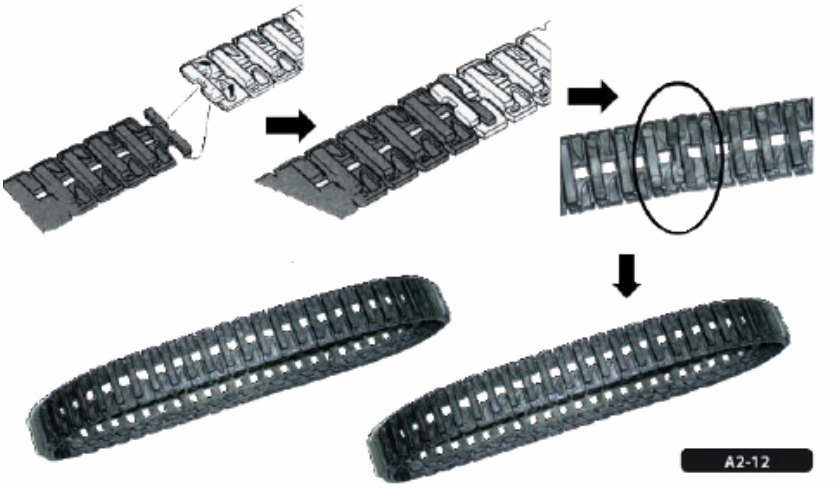


Рисунок 1.33 – А2.12

Оденьте гусеницу на направляющие колеса робота (см. рис. 1.31 - А2.10).

Рисунок 1.34 – А2.13

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 (см. рис. 1.32 - A2.11).

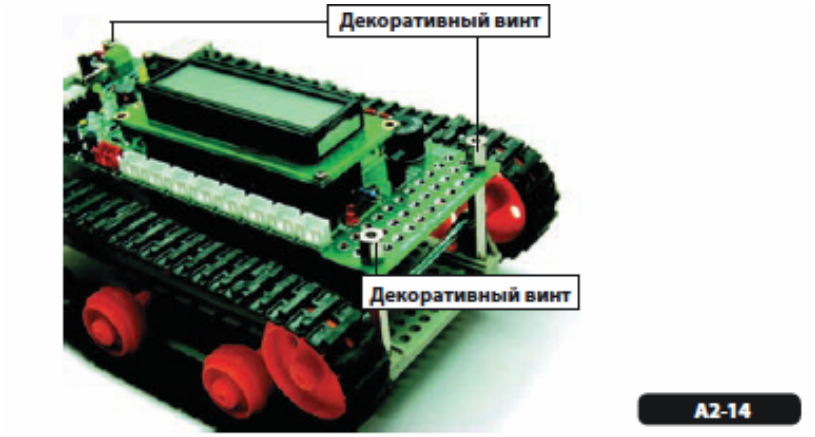


Рисунок 1.35 – A2.14

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 (см. рис. 1.33 - A2.12).

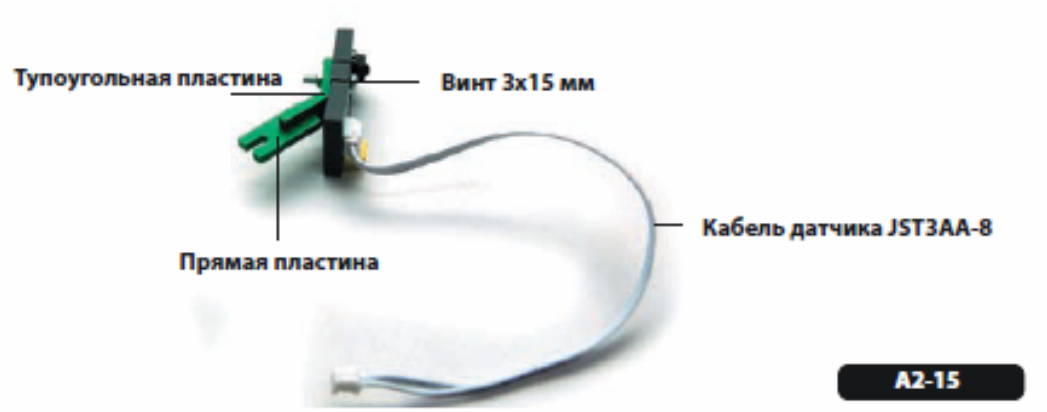


Рисунок 1.36 – A2.15

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 (см. рис. 1.34 - A2.13).

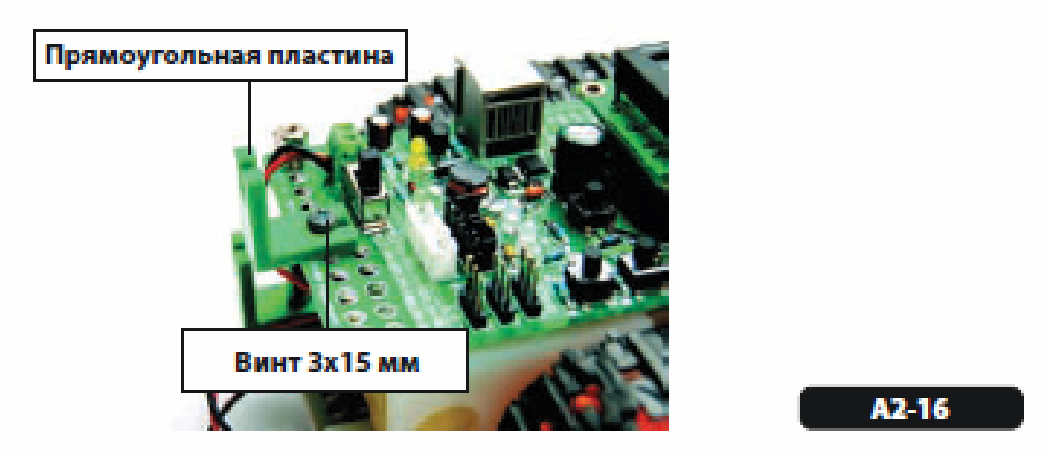


Рисунок 1.37 – A2.16

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 (см. рис. 1.35 - A2.14).

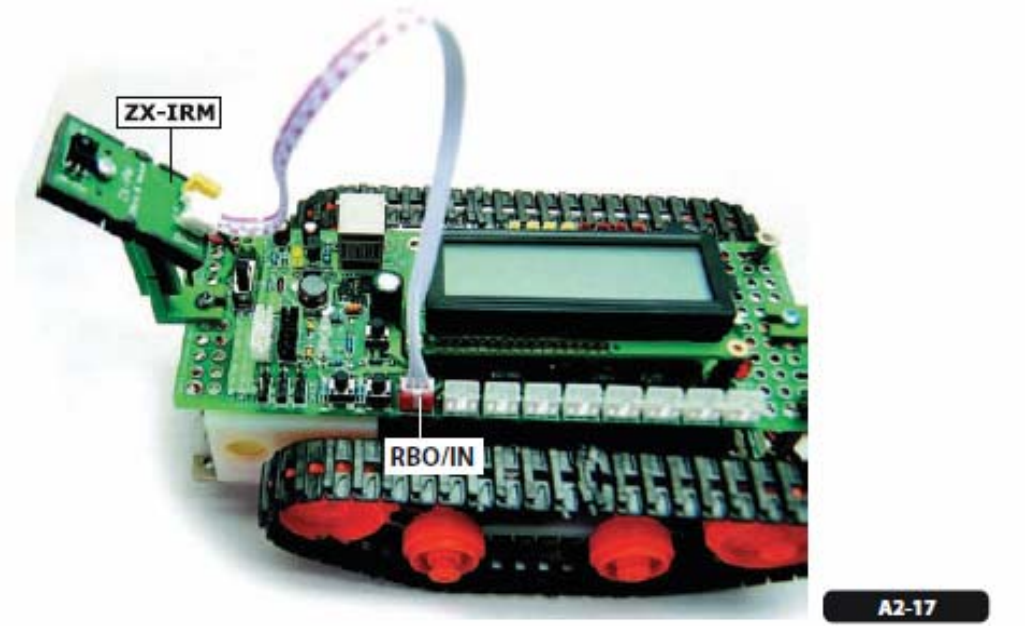


Рисунок 1.38 – A2.17

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 – 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, когда оба светодиода светятся зеленым светом, это означает,

что происходит движение вперед; если же 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 (см. рис. 1.36 - A2.15).

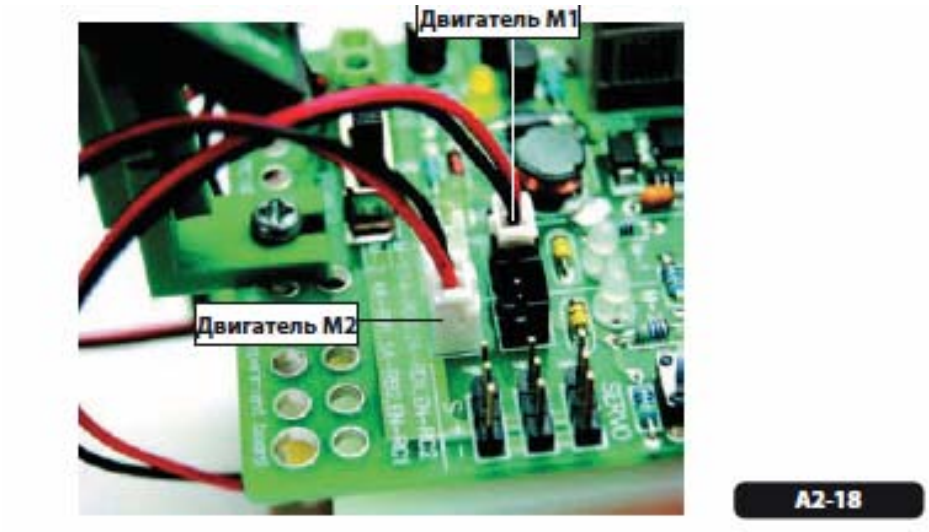


Рисунок 1.39 – A2.18

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-миллиметровую пластиковую шайбу, пластину и закрепив 3-миллиметровой гайкой. Установите обе платы: одну – с правой стороны корпуса робота, другую – с левой стороны (см. рис. 1.37 - A2.16).

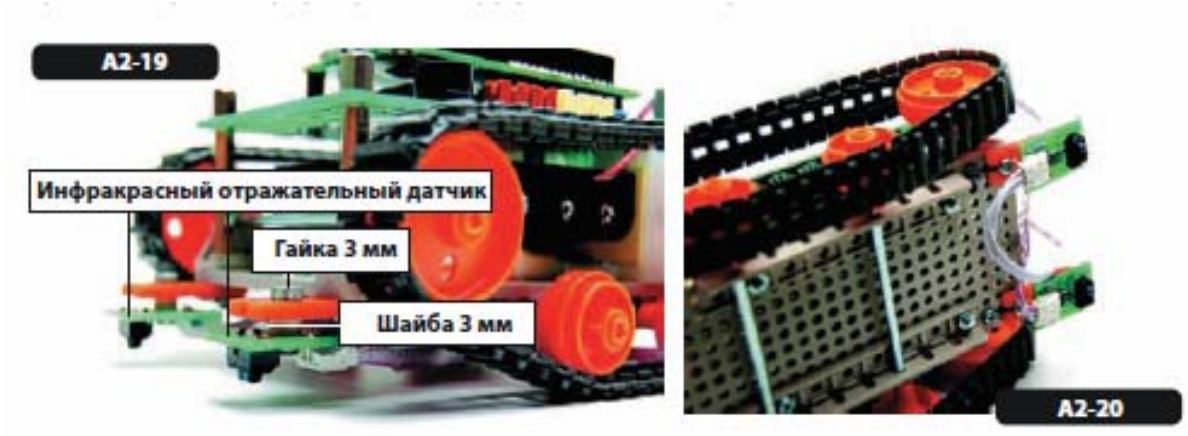


Рисунок 1.40 – A2.19, A2.20

Соедините модуль GP2D120 с прямоугольной пластиной (см. рис. 1.38 - A2.17, рис. 1.41 - A2.21), используя винт 3x10 мм и гайку 3 мм.

Рисунок 1.41 – A2.21

Спереди робота вставьте винт 3x10 мм в центральное отверстие платы RBX-877V2.0 и гайку 3 с верхней стороны платы (см. рис. 1.39 - A2.18, рис. 1.42 - A2.22). Не затягивайте винт. Далее вставьте модуль GP2D120, собранный на шаге A2.17, между головкой винта и платой управления (см. рис. 1.42 - A2.23). Затяните винт для полной фиксации.

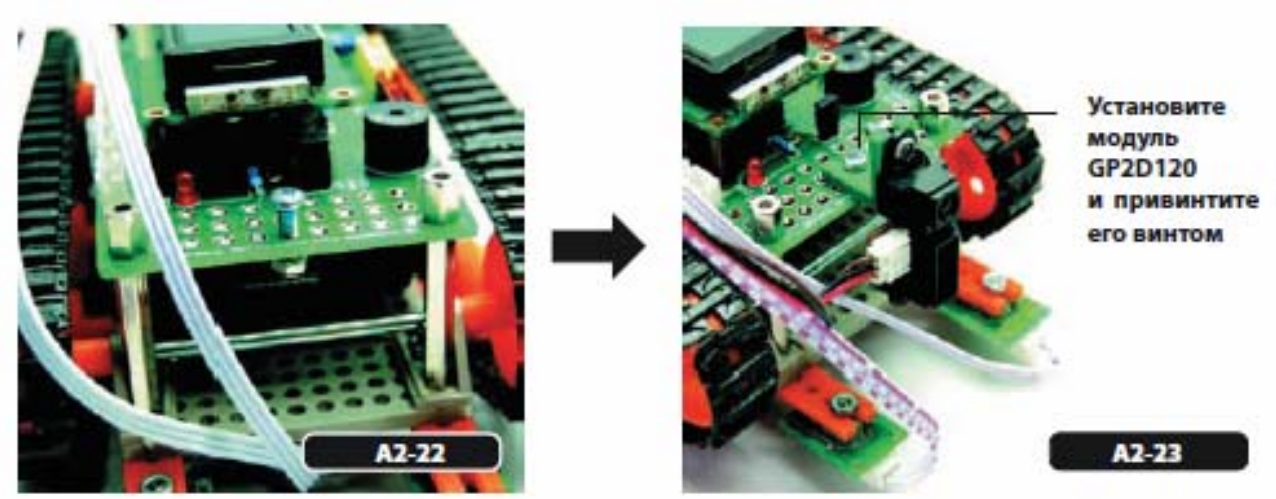


Рисунок 1.42 – A2.22, A2.23

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 (см. рис. 1.40 - A2.19).

Подровняйте все кабели и проверьте правильность всех соединений (см. рис. 1.40 - A2.20, рис. 1.43).

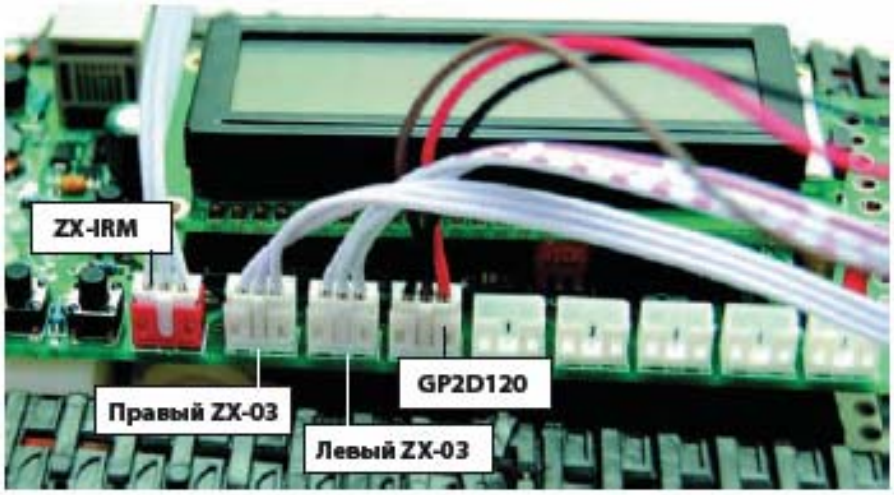


Рисунок 1.43 – Подключение всех соединений

Теперь Ваш Robo-51 готов к программированию (см. рис. 1.44).

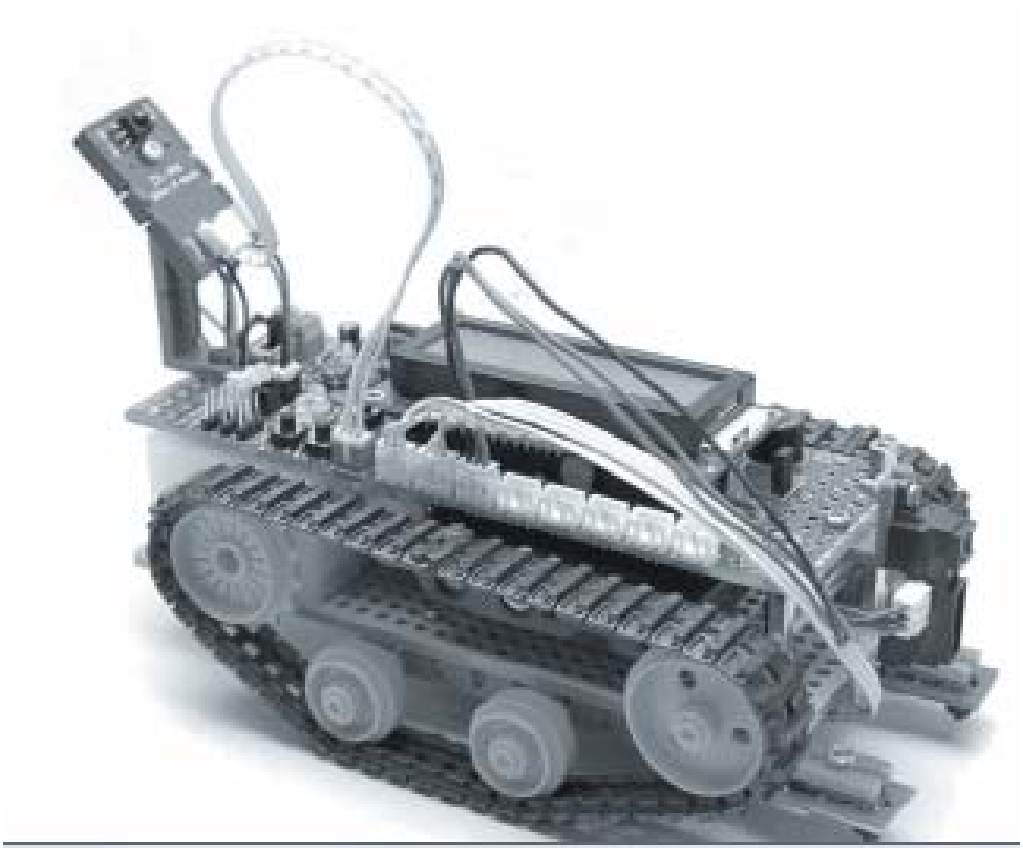


Рисунок 1.44 – Собранный лабораторный стенд

Задание по выполнению лабораторной работы:

1. Изучить основные составляющие части лабораторного стенда.
2. Выполнить сборку лабораторного стенда согласно пошаговому описанию.

Контрольные вопросы:

1. Понятие мехатроники и мехатронных систем.
2. Базовые объекты изучения мехатроники.
3. Состав мехатронных систем.

Лабораторная работа №2

Написание программы для тестирования платы управления RBX-877 V2

Цель работы:

Изучить плату управления мехатронной системой IE-ROBO-51, написать программу для её тестирования.

Основные теоретические сведения:

Плата управления RBX-877 V2 построена на базе процессора с гибким программированием T89C51AC2.

Описание и особенности микросхемы

Особенности:

- 80C51 основная архитектура;
- 256 Байтов оперативной памяти On-chip;
- 1 Кбайт On-chip XRAM;
- 32 Кбайт флеш-памяти On-chip
  Срок хранение данных: до 10 лет в 85C
  Цикл Read/Write: 10K;
- 2 Кбайт флеш-память On-chip для загрузочног сектора;
- 2 Кбайт On-chip EEPROM;
  Цикл Read/write: 100K;
- 4-горизонтальных прерывания 14-источников данных;
- Три 16-разрядных Таймера;
- Полный спаренный UART совместимый с 80C51;
- Максимальная частота для кристалла - 40 Мгц,
  в X2-методе - 20 Мгц (Ядро ЦП, 20 Мгц);
- Пять портов: 32 + 2 цифровых линии I/O;
- Пятиканальный 16-разрядный PCA;
- Двойной указатель данных;
- 21-разрядный таймер(7 Программируемых);
- On-chip логика;
- Энергетические методы сохранения данных:
  Обычный метод
  Метод Power-down;
- Электроснабжение: 3V к 5.5V;
- Рабочая температура: - 40C до +85C;
- Корпуса исполнения: VQFP44, PLCC44;

Описание:

T89C51AC2 - высоко производительная версия 80C51 с 8-ми разрядным микроконтроллером. Содержит 32 Кбайта флеш-памяти для программы и данных.

32 килобайта флеш-памяти могут программироваться или в параллельном режиме, или в серии режима со способностью ISP или программным обеспечением. Внутреннее напряжение программирования производится от стандартного вывода VCC.

T89C51AC2 сохраняет все особенности 80C51 с 256 байтами внутренней оперативной памятью, 7-исходный 4-горизонтальный диспетчер прерывания и три порта таймера/счетчика. Кроме того T89C51AC2 имеет 10-разрядный АЦП конвертер, 2 Кбайта загрузочной флеш-памяти, 2 Кбайта EEPROM для данных, программируемого массива счетчика, XRAM на 1024 байтов, аппаратный Watch-Dog Timer и более многосторонний серийный канал, который облегчает коммуникацию (EUART) мультипроцессора. Полностью статический дизайн T89C51AC2 сокращает энергетическое потребление системы, сбивая частоту к любому значению без потери данных, даже таких, как DC.

T89C51AC2 имеет два выбираемых программным обеспечением метода уменьшенной деятельности и 8-разрядные часы для дальнейшего снижения в энергетическом потреблении. В режиме холостого хода ЦП блокируется, пока внешние устройства и система прерывания все еще действуют. В режиме Power-down оперативная память сохранена, и все остальные функции являются недействующими.

Дополнительные особенности T89C51AC2 делают его мощнее для приложений, где нужно A/D (АЦП) преобразование, импульсной модуляции, высокой скорости I/O и рассчитывающей способности, как например технический контроль, товары широкого потребления, сигнализации, моторный контроль и др.

Оставаясь полностью совместимым с 80C52, T8C51AC2 предлагает супернабор стандартного микроконтроллера. В режиме X2 максимальная внешняя тактовая частота составляет 20 мгц, а время цикла достигает 300 нсек.

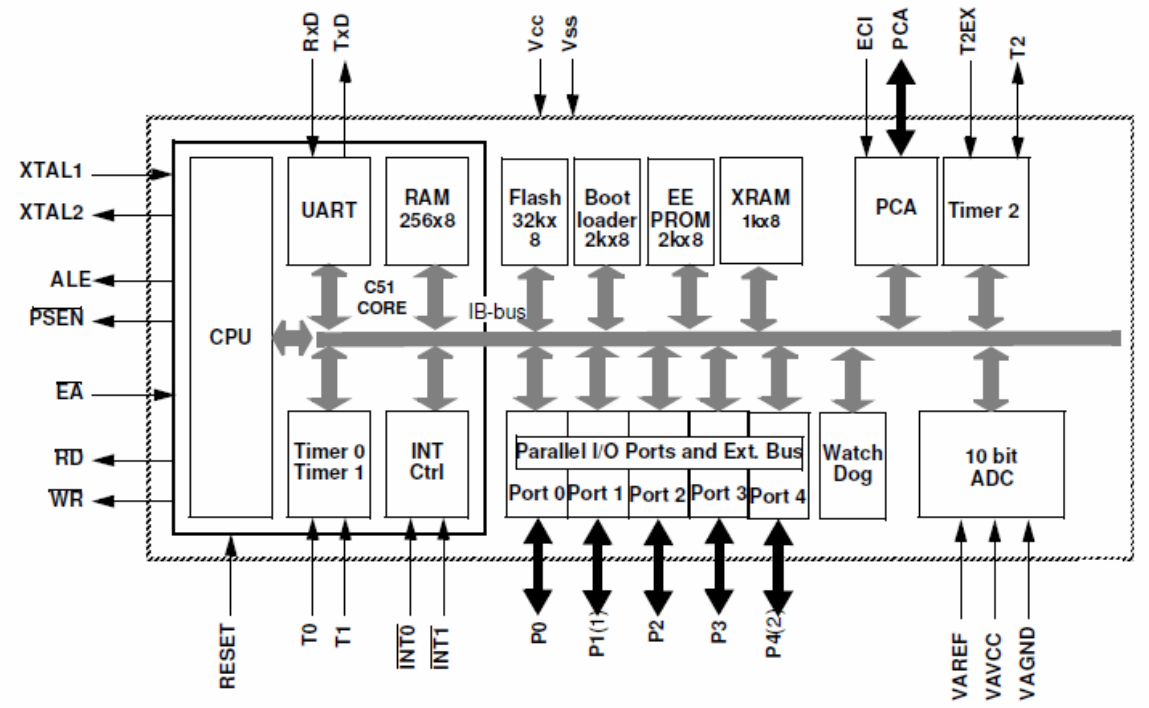
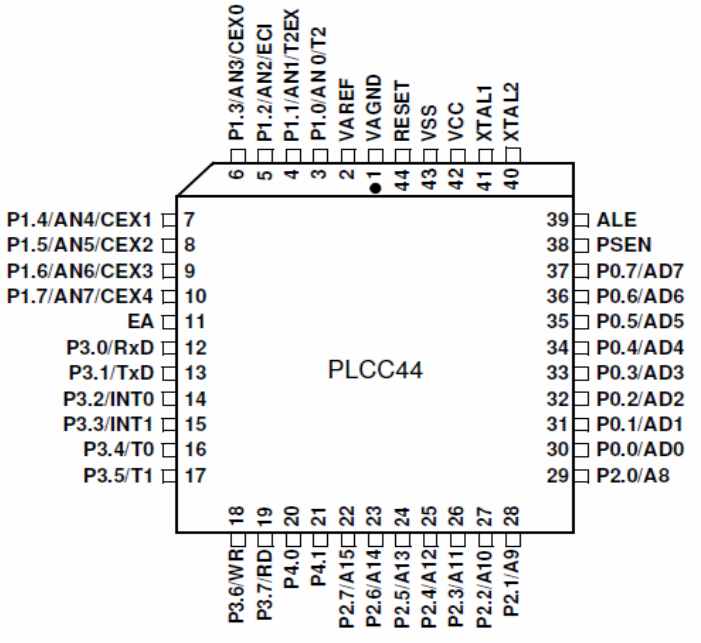


Рисунок 2.1 – Блок-схема T89C51AC2



а

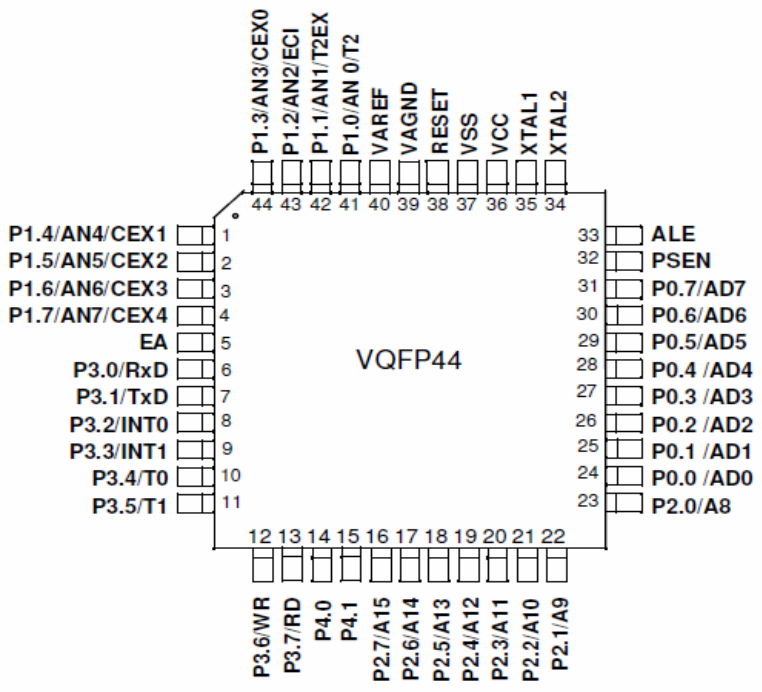


Рисунок 2.2 – Конфигурация выводов
а) для PLCC4
б) для VQFP44

Таблица 2.1 – Конфигурация выводов

Имя	Тип	Вывод
VSS	GND	Схемная земля
VCC		Напряжение питания
VAREF		Опорное напряжение VAREF для ADC
VAGND		Опорная земля для ADC
P0.0:7	I/O	8-разрядный открытый дренажный двусторонний порт ввода/вывода. Порт 0 вывода имеет собственные колебания, и в этом состоянии может быть использован как выход высокого импеданса. Порт 0 имеет также мультиплексные свойства младшего адреса, шину данных доступа к внешней программе и память данных.

P1.0:7	I/O	8-разрядный двусторонний порт ввода/вывода с внутренними управлениями. Выводы 1 порта могут быть использованы для цифрового входа/выхода или как аналоговые входы для конвертера (ADC). Выводы порта 1 снабжены высокоомными внутренними транзисторами каскадами и могут быть использованы в этом состоянии как входы. Выводы порта 1 предназначены для использования аналоговых входов через регистр ADCCF (в этом случае внутренние каскады разъединены). Как вторичная цифровая функция порт 1 содержит таймер внешнего триггера и вход генератора тактовой частоты; PCA внешний вход генератора тактовой частоты и PCA модуль I/O (ввода-вывода). P1.0/AN0/T2 Аналоговый вход канала 0, Внешний вход генератора тактовой частоты для Таймер/счетчика. P1.1/AN1/T2EX Аналоговый вход канала 1, Вход триггера для таймера/счетчика. P1.2/AN2/ECI Аналоговый вход канала 2, Внешний вход генератора тактовой частоты PCA. P1.3/AN3/CEX0 Аналоговый вход канала 3, Модуль ввода PCA 0 вход/PWM выход. P1.4/AN4/CEX1 Аналоговый вход канала 4, Модуль ввода PCA 1 вход/PWM выход. P1.5/AN5/CEX2

		Аналоговый вход канала 5, Модуль ввода PCA 2 вход/PWM выход. P1.6/AN6/CEX3 Аналоговый вход канала 6, Модуль ввода PCA 3 вход/ PWM выход. P1.7/AN7/CEX4 Аналоговый вход канала 7, Модуль ввода PCA 4 вход/ PWM выход. Порт 1 получает младший адресный байт в течение стираемой программируемой постоянной памяти и верификация программы. Это может управлять входами CMOS без внешних схем.
P2.0:7	I/O	- 8-разрядный двусторонний порт ввода/вывода с внутренними каскадами. Выводы порта 2 имеют внутренние каскады и могут быть использованы как входы в этом состоянии. В течение времени доступа к внешним данным памяти порт пользуется 8 разрядными адресами (MOVX @Ri), порт 2 передает содержимое из специального функционального регистра. Он также получает старшие адреса и управляет сигналами в течение программного опроса. Порт может управлять входами CMOS без внешних схем.
P3.0:7	I/O	8-разрядный двусторонний порт ввода/вывода с внутренними каскадами. Выводы порта 3 могут быть использованы в этом состоянии как входы. Выход ключа, соответствующий вторичной функции, должен программироваться одинаково, чтобы эта функция действовала (за исключением TxD и WR). Вторичные функции предназначены выводам порта 3 указаны ниже:

		P3.0/RxD: Ввод (асинхронный) данных получателя или данные (синхронные) ввода/вывода последовательного интерфейса. P3.1/TxD: Передатчик данных выхода (асинхронный) или данные (синхронные) генератора тактовой частоты последовательного интерфейса. P3.2/INT0: Внешнее прерывание 0 входа/таймера 0 данных контрольного входа. P3.3/INT1: Внешнее прерывание 1 входа/таймера 1 данных контрольного входа. P3.4/T0: Таймер 0 встречного входа P3.5/T1: Таймер 1 встречного входа P3.6/WR: Внешняя память данных записывает строб; запирает байт данных от порта 0 во внешней памяти данных. P3.7/РЕЗЕРФОРДА: Внешняя память данных читает строб; разрешает внешнюю память данных. Она может управлять входами CMOS без внешних каскадов.
P4.0:7	I/O	2-разрядный двусторонний порт ввода/вывода с внутренними каскадами. Выводы порта 4, могут быть использованы как входы в этом состоянии. Как входы, выводы порта 4, будут текущим источником из-за внутреннего транзистора каскада

RESET	I/O	Установка сброса: Высокий уровень на этом выводе в течение двух циклов, пока осциллятор работает снова устанавливает устройство в исходное состояние. Внутренний резистор присоединенный к VSS разрешает сброс включенного питания, пользуясь только внешним конденсатором присоединенным к VCC.
ALE	O	Адресный ключ разрешает состояние для запирания младшего байта адреса в течение доступов к внешней памяти. ALE является активной каждые 1/6 периодов (1/3 в X2 режиме) осциллятора, кроме доступа памяти внешних данных. Когда инструкция выполняется от внутреннего Flash (EA=1), ALE может блокировать программное обеспечение.
PSEN		Программное хранение включения выхода является контрольным сигналом, который разрешает внешнюю программную память шины в течение времени полученных внешних операций. Оно активизируется дважды каждого аппаратного цикла во время выборки от внешней программной памяти. Однако, когда выполняется внешней программной памятью, две активации PSEN пропускаются в течение каждого доступа к внешним данным памяти.
EA	I	Когда внешний доступ проводится на высоком уровне, инструкции выбираются из внутренней флэш-памяти, когда программа считывает менее 8000H. Когда проводится на низком уровне – A/T89C51AC2 получает все инструкции из внешней программной памяти.

XTAL1	I	Вход инвертора усилителя осциллятора и выход внутренних импульсов тактового генератора. Чтобы управлять устройством от внешнего тактового источника, XTAL1 можно управлять, пока XTAL2 остается несвязанным. Для действий выше частоты 16 Мгц, нужно поддерживать 50% рабочий цикл.
XTAL 2	I	Выход для инвертирующего усилителя осциллятора

Средства разработки для программирования роботов

В комплекте Robo-51 используется микроконтроллер MCS-51, T89C51AC2, программа может быть написана на языках программирования BASIC и C++. Программы BASIC и C++ должны использовать компилятор программного обеспечения.

Однако в этом комплекте представлена программа C++ во всех примерах. Компилятор C++ используется в Raisonance 8051 с именем Rkit-51. Этот инструмент включает в себя множество программного обеспечения для поддержания развития программирования на языке C++, таких как RIDE - IDE для создания кода в C++, RC51 – C++ компилятор, MA51 - асемблер, LX51 - Linker инструменты, LIB51 - менеджер библиотек.

Для этого робота выбрана демо-версия Rkit-51. Предельный размера кода составляет 4 КБ.

Пользователи могут приобрести полную версию от Raisonance на сайте www.raisonance.com. Однако комплект в Robo-51 содержит это программное обеспечение в комплекте CD-ROM.

Пользователи могут увидеть инструкции из Rkit-51 и все соответствующие программные средства после установки. В данном руководстве нет детальной инструкции.

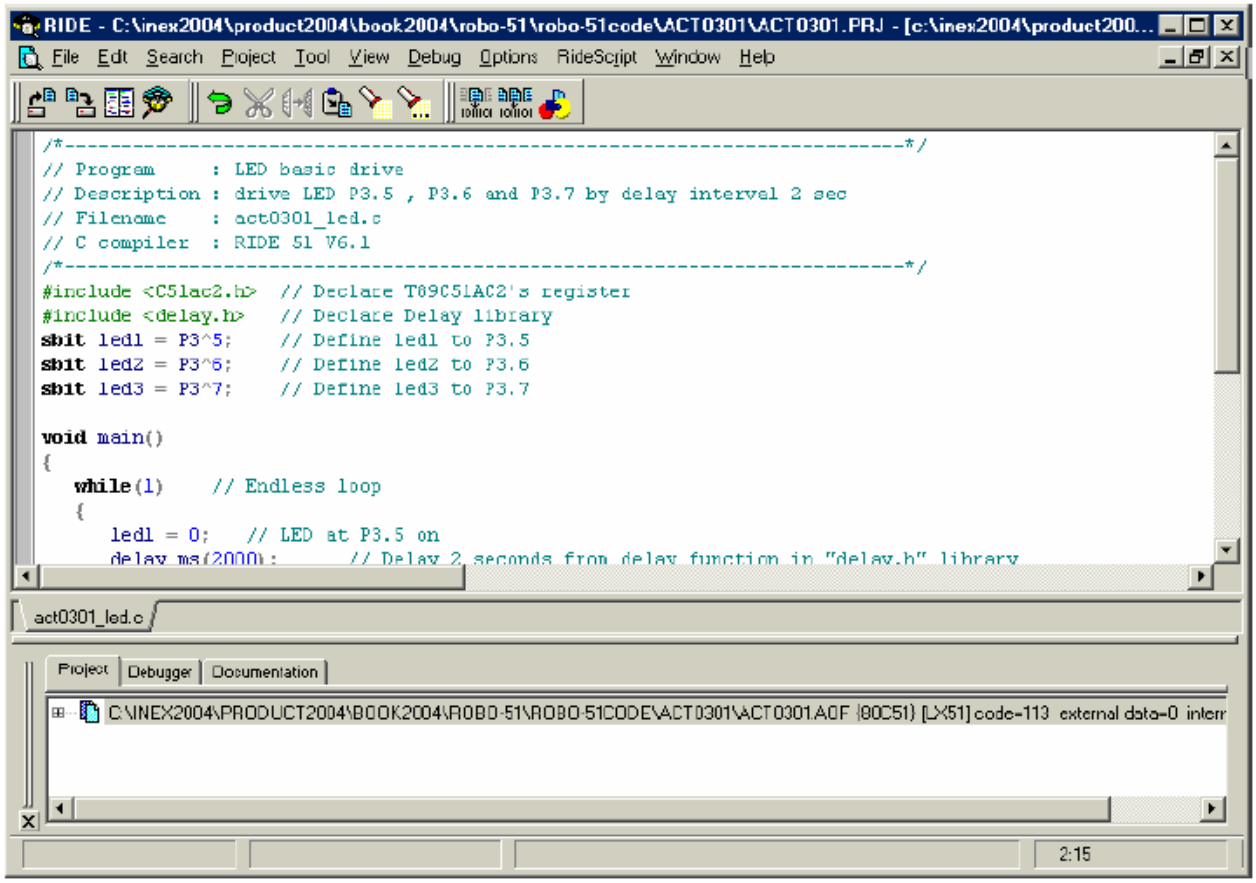


Рисунок 2.4 – Окно RIDE из средств разработки программ для роботов Rkit – 51

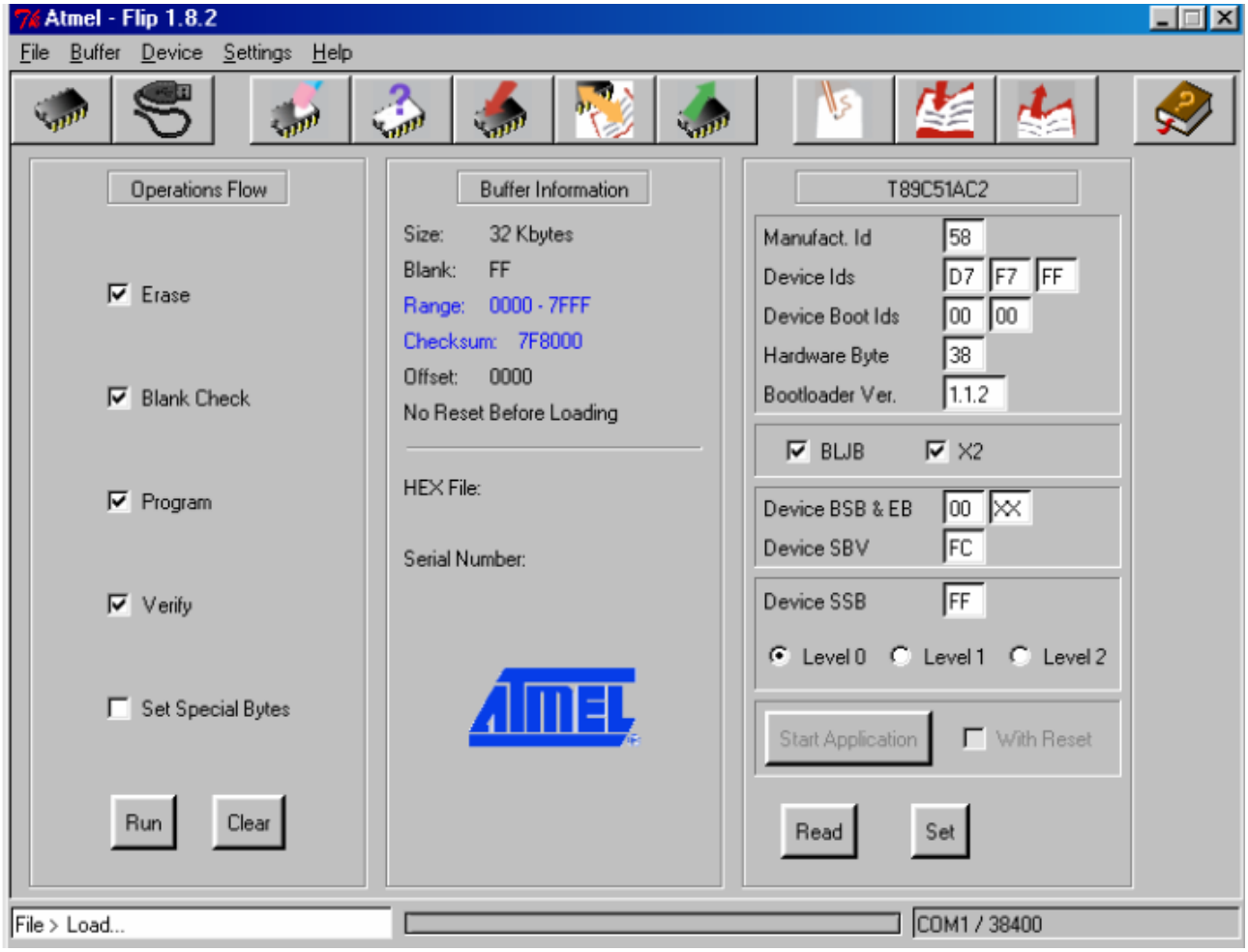


Рисунок 2.5 – Flip из средств разработки программ для роботов Rkit – 51

После компиляции C++ в результате получите файлы HEX (в формате .hex). Необходимо загрузить этот код в память программы, в микроконтроллер T89C51AC2.

Загрузите программное обеспечение FLIP от Atmel Corporation. Скачать бесплатно его можно на сайте www.atmel.com.

Введение инструментальных программных средств для комплекта Robo- 51

Введение для Rkit- 51

Robo-51 - это образовательный робототехнический комплект, программирование которого осуществляется на языке C++. Инструмент развития - Raisonance 8051 от французских разработчиков;
В комплекте Raisonance S.A. имеется набор программного обеспечения под названием Rkit- 51. Особенности этого комплекта:

- лучшее выполнение функций среди профессиональных инструментов;
- превосходная кодовая плотность (лучше, чем CodeCompressor (TM));
- оптимизирующий компилятор C++;
- интуитивный IDE (одно обращение к программе, чтобы запустить симулятор и загрузить программу);
- мощный IDE (Моделирование и Периферийный Dev. Кит);
- точные симуляторы (представление разрядного уровня UART)
- техническое обеспечение и программное обеспечение от IDE (нет проблем с совместимостью);
- поддержка подсказок (помощь и подсказки);
- поддерживает много номеров MCS- 51 от лидера изготовителя, как например AT89xxx от Atmel, P89C51Rxx и P89LPC9xx или ADuC8xx и т.п.;
- обеспечивает всестороннюю версию оценки. Пользователь может проверить и оценить действие коммерческой версии. Размер кода ограничен - 4КБайта.

Для более подробной информации рекомендуется использовать сайт www.raisonance.com.

Установка инструментов для Rkit- 51

Rkit-51 объединяет 5 инструментальных программных средств под RIDE (Raisonance Integrated Development Environmen) - Объединенное Развитие Окружающей среды. В переводе с англ. RIDE – «езда».

RIDE - окно программы, которое позволяет пользователю легко создавать проекты, называть проект, вызывать Компилятор, Ассемблер и Редактора связей, чтобы строить проект, симулировать, или выполнять отладку. Список инструментов, включаемых в комплект Rkit- 51 с кратким обзором:

- (1) Компилятор: RC-51 ANSI C++ приводит код C++ к коду Ассемблера;
- (2) Ассемблер: MA- 51 превращает код программы в машинный код.

- (3) LX-51 Редактор связей: комбинирует объектные файлы, производимые близко Компилятору и Редактору связей, и производит объектный файл. Редактор связей также решает, где определенные виды Данных и Кода будут расположены в памяти.

- (4) LIB-51 Менеджер библиотек берет объектные файлы, производимые Компилятором или Ассемблером, и создает библиотеку;

- (5) OH51XA – шестнадцатеричный для объекта Конвертер – превращает объектный файл, подобно редактору связей и делает шестнадцатеричный файл.

На рис. 2.6 показано соотношение между инструментами.

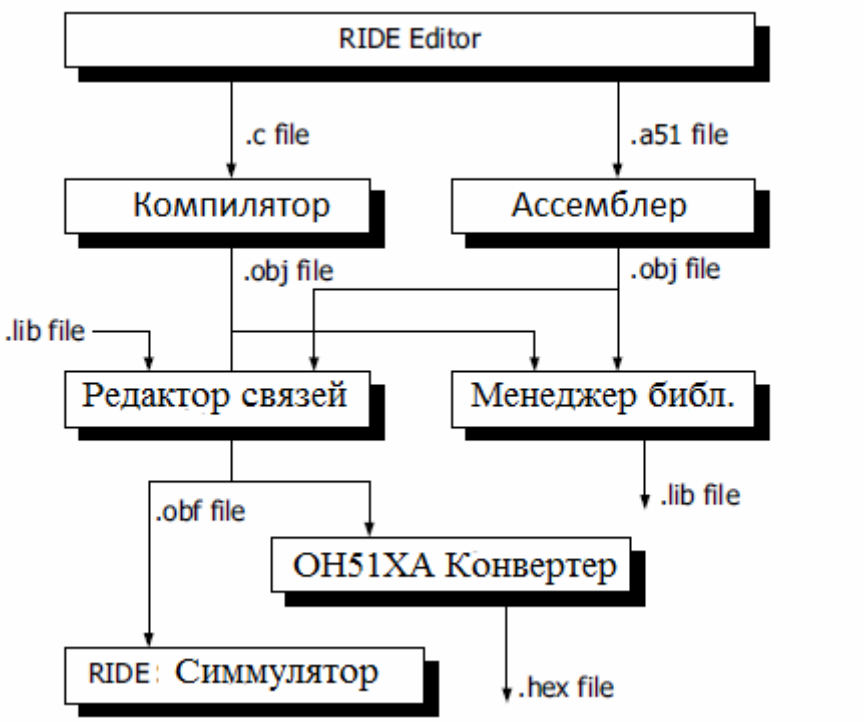


Рисунок 2.6 – Диаграмма соотношений между инструментами для Rkit- 51

Установки

Если программное обеспечение устанавливается через КОМПАКТ-ДИСК, то инсталляционная программа должна автоматически установиться при вставленном КОМПАКТ-ДИСКЕ.

Если вы загрузили программное обеспечение от веб-узла, то возможно программное обеспечение установится просто запуская INSTALL.EXE.

Минимальные Системные Требования для Rkit- 51:

- Windows 98/NT/2000/XP;
- Pentium Processor;
- 30Mb Hard Drive Space;
- 64Mb RAM.

Запуск Rkit- 51

Запустить RIDE очень легко. Выберите «Ride IDE» меню Start→Programs→Raisonance→Kit. Программа выдаст следующее окно:

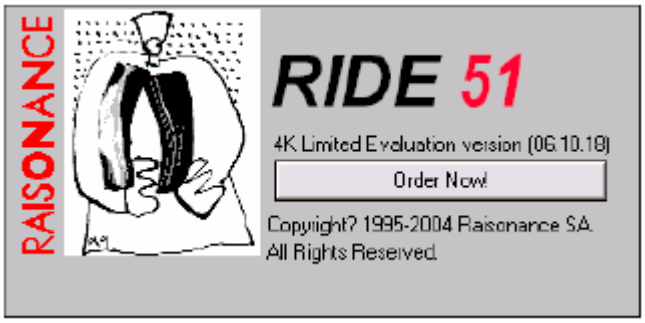


Рисунок 2.7 – Окно выдачи при запуске RIDE

Через несколько минут главное окно, описание которого будет представлено в дальнейшем, откроется.

Создание проекта

Первое, что нужно сделать, это создать новый проект. Посмотрите на главное окно RIDE и ознакомьтесь с ним.

(1) Выберите Project→New, у вас появится окно, подобное следующему:

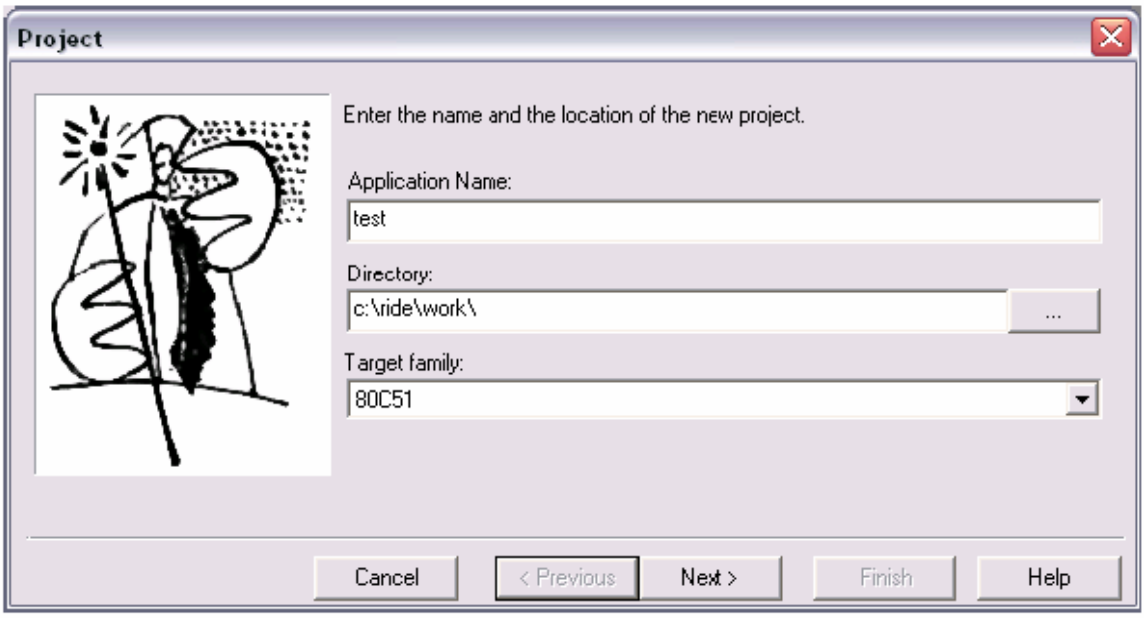


Рисунок 2.8 – Окно 1 для создания проекта в RIDE

(2) поле имени показывает путь проектному файлу, который будет создан. Тип файла отображает семейство микроконтроллера, тип проекта, который будет использоваться. В зависимости от комплекта разработчика поле Type покажет также: 80C51, XA или ST6. Здесь выберите 80C51.

(3) Щелкните кнопку Просмотр (Browse**)** и рассматривайте папки, где будет создан проект. Пример будет расположен: C:\ride\work.

(4) В поле Имя файла имя проекта введено. Введите test и щелкните следующую кнопку. Целевое окно выбирает целевой микроконтроллер.

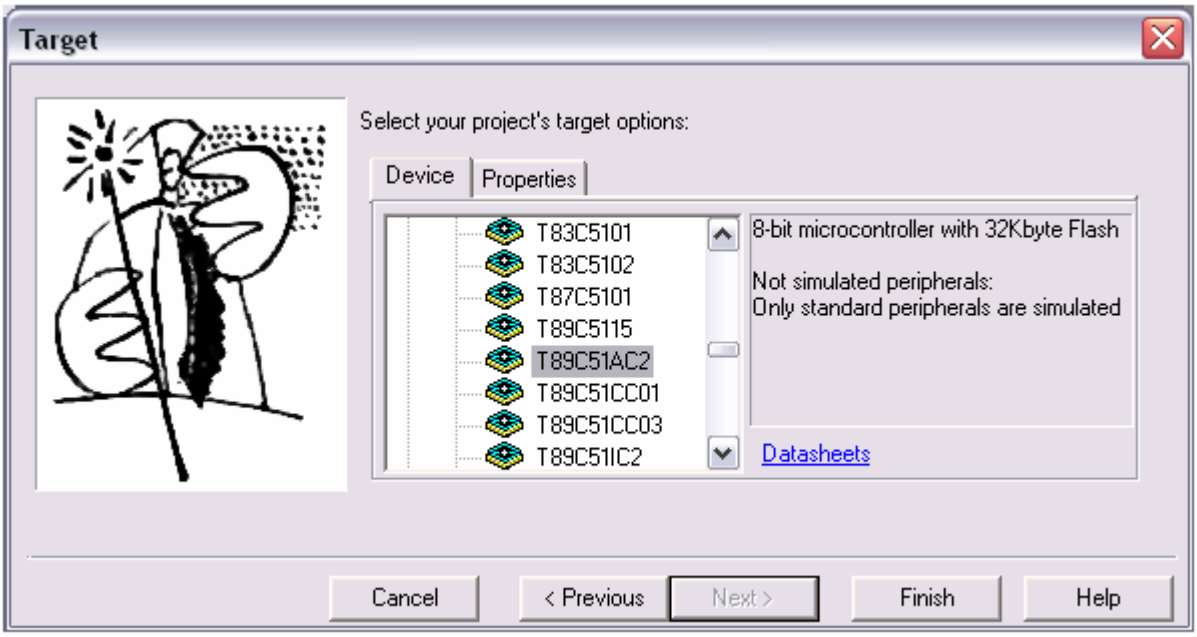


Рисунок 2.9 – Окно 2 для создания проекта в RIDE

(5) В перечне Устройств, выбирают Atmel _ T89C51AC2. После этого выбирают Properties и Harvard Architechture. Щелкните кнопку «Закончить» (Finish), чтобы создать проект. Окно RIDE должно сейчас показать следующее:

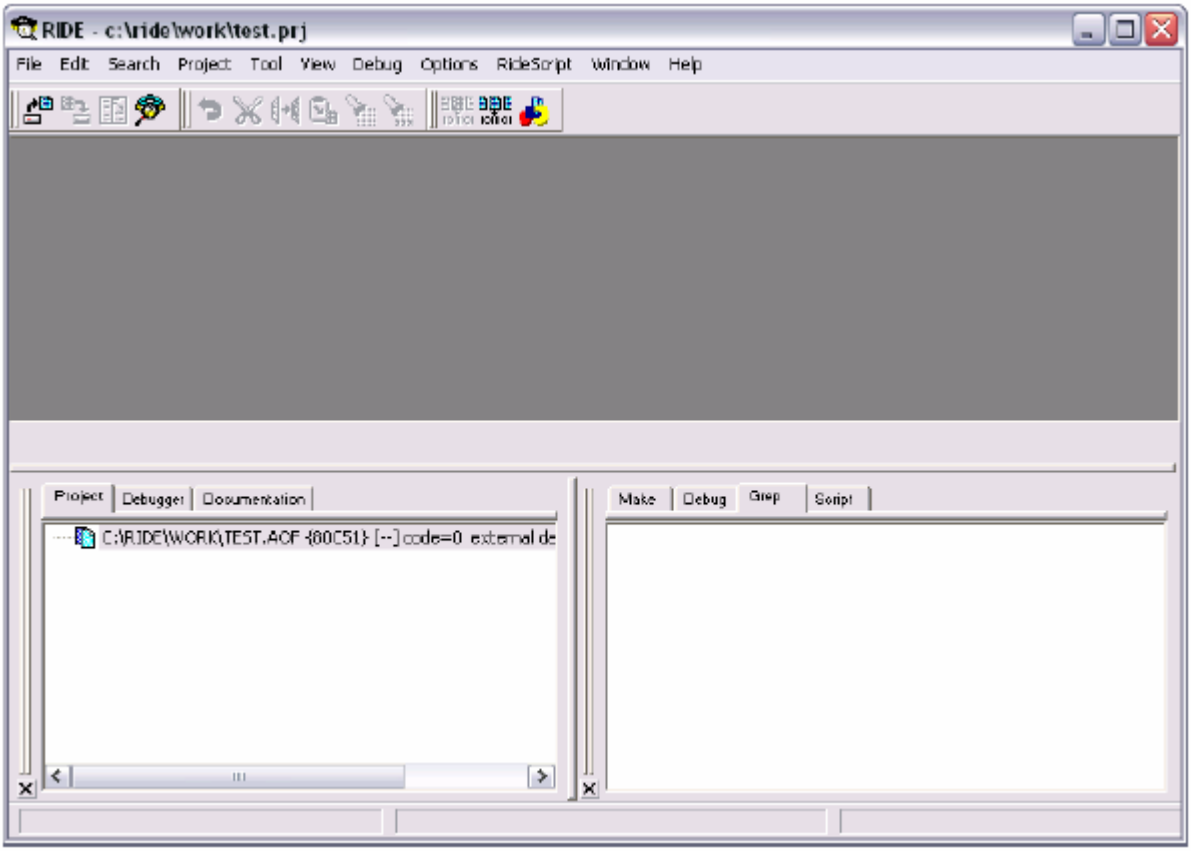


Рисунок 2.10 - Окно 3 для создания проекта в RIDE

Проектное окно действует подобно проектному менеджеру, показывая источник файлов, находящихся в проекте и предоставляя мгновенный доступ каждому из них: как редактору, так и отладчику. Если смотреть на проектное окно, то вы будете видеть один вход со следующим именем пути: C:\РАБОЧИЙ\TEST.AOF. Этот вход представляет проект как целое и .aof файл будет результатом строительства проекта.

Создание и добавление исходного файла

Следующий шаг - создание нового, основного исходного файла и добавление его в проект. Далее будет показано, как построить проект.

Чтобы создать новый исходный файл выбирают File→New завершено “C Files” для всплывающего меню, которое появится. Откроется пустое окно.

Введите следующее в новое окно:

```
void main (void)
{
while (1) ;
}
```

Чтобы сохранить исходный файл:

1) Выберите File→Save As. Стандартное сохранение, окно откроется.
2) Введите main.c в поле Имя файла.
3) Щелкните «Сохранить».

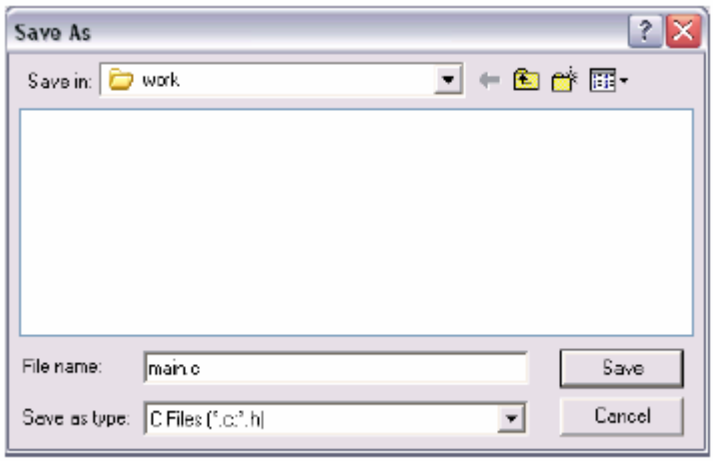


Рисунок 2.11 - Окно 1 для создания файла в RIDE

Чтобы добавить файл к проекту:

1) К меню Project→Add note Source/Application или Alt + Insert ;

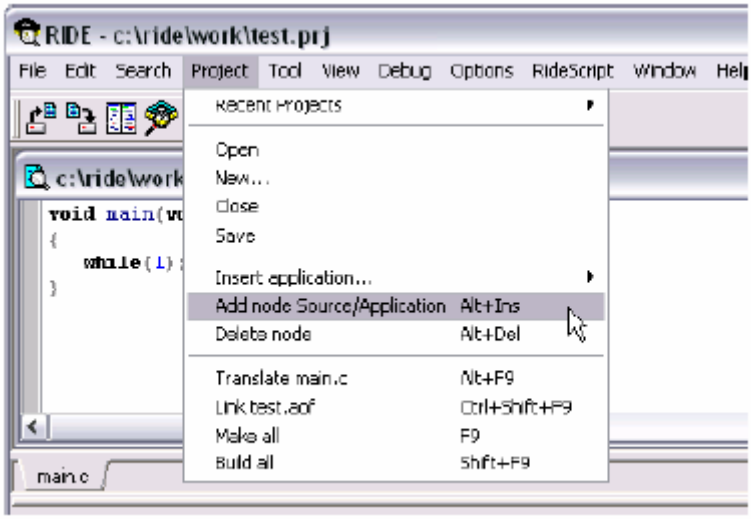


Рисунок 2.12 - Окно 2 для создания файла в RIDE

2) Выберите файл, который требуется для добавления (main.c). Щелкните кнопку «Открыть» (Open).

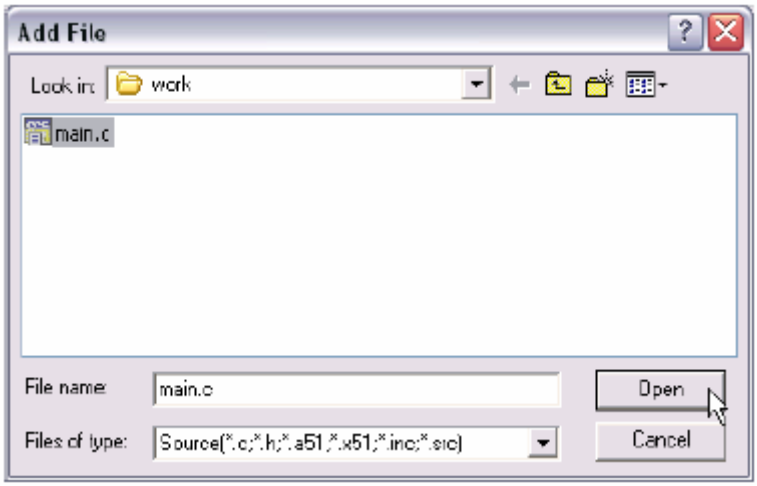


Рисунок 2.13 - Окно 3 для создания файла в RIDE

3) В окне проекта должен появиться исходный файл .aof:

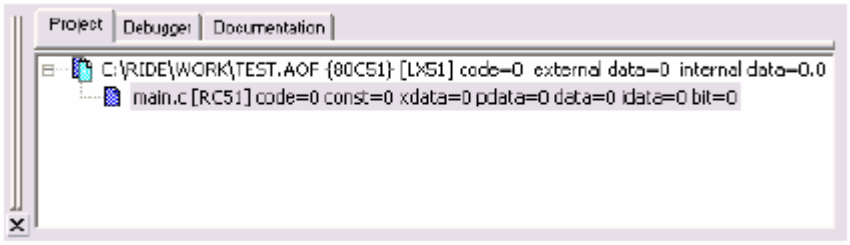


Рисунок 2.14 - Окно 4 для создания файла в RIDE

Построение проекта

Чтобы построить проект, просто щелкните по кнопке Make All на панели инструментов или выберите Project→Make All;

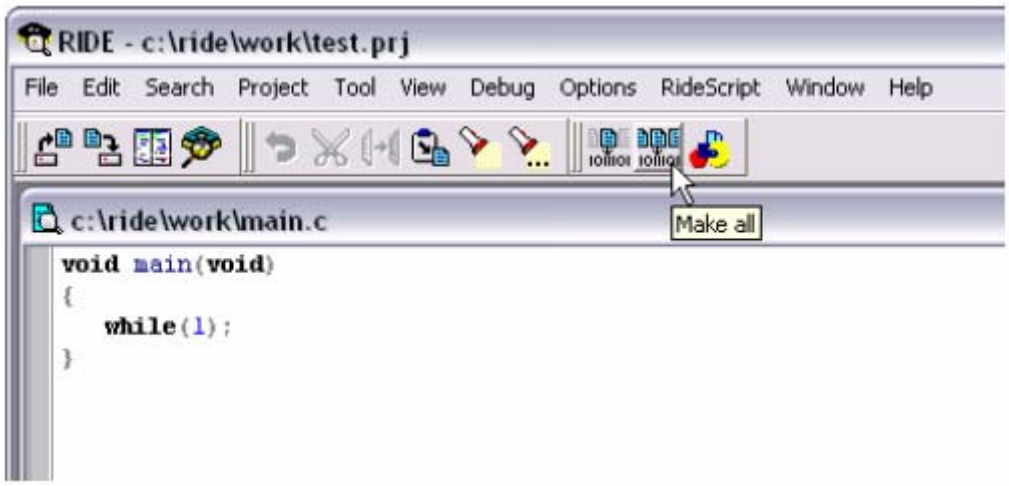


Рисунок 2.15 - Окно 1 для построения проекта в RIDE

Как только проект построится, окно Make покажет результат:

Рисунок 2.16 - Окно 2 для построения проекта в RIDE

Как только проект успешно построен, все готово к дальнейшим действиям.

Введение для Flip

Flip (с англ. «щелчок») - гибкое приложение ПК, которое позволяет вам создать программу и конфигурацию Atmels. Эта новая главная версия Flip предлагает следующее:

- выполняет системное программирование для входа через RS232, USB или интерфейсы;
- может быть использована через интуитивный графический интерфейс пользователя или запускаться из окна DOS, встроенное программное обеспечение IDE подобно KEILs u Vision2, или даже вашему собственному приложению (см. ISP Functions Library);
- совместима с Windows 9x / Me / NT / 2000 / XP;
- поддерживает Intel MCS-86 Hexadecimal Object, файл Code 88 для форматирования данных, загрузки и сохранения;
- способности редактирования буфера: заполнение, поиск, копия, сброс, изменение, память устройства;
- управление памятью: стирание, пустая проверка, программа, проверка, чтение, уровень безопасности и специальные байты, читающие и устанавливающие условия технического обеспечения;
- ISP возможна установка программным обеспечением.
- демонстрационный метод подражает действиям ISP без какого-либо целевого технического обеспечения.

Подходящая версия FLIP для эксперимента Robo-51 - V1.8 или выше. Поскольку эта версия будет поддерживаться микроконтроллером T89C51AC2. Функция FLIP загружает шестнадцатеричный код (HEX) от компьютера к микроконтроллеру T89C51AC2 на плате управления RBX-51AC2. Эта плата управления является ядром управления для Robo-51.

Установка

Если программное обеспечение устанавливается через компакт-диск, то инсталляционная программа должна автоматически установиться при вставленном компакт-диске. Найдите загрузочный файл setup.exe.

Если компакт-диск выключен или вы загрузили программное обеспечение из веб-узла, то программное обеспечение может установиться просто запуском файла setup.exe.

После установки комплекта: Programs→Atmel→Flip 2.2.4→Flip (число версии может изменяться).

Запуск Flip

После того как установка Flip закончится, появится главная составляющая. Посмотреть всю деталь можно в фигуре 2-1.

В главном окне FLIP, можно увидеть три рамки слева направо:

- Operations flow - вытекающие операции;
- Buffer information - информация буфера;
- Device parameters - параметры устройства.

Окно сообщений, область прогресса и информация коммуникации представлены внизу главного окна.

Важное командное меню

File

Load Hex file : Выбор файла HEX для загрузки.

Save Hex file : Хранение текущих шестнадцатеричных данных (HEX) в буфере к HEX файлу.

Buffer

Edit : Проверка и изменение данных в буфере.

Option : Установка параметра выбора в буфер.

Device

Select : Выбор целевого микроконтроллера.

Erase : Стирание всех данных в показном участке целевого микроконтроллера.

Blank check : Проверка памяти.

Read : Чтение или получение данных из микроконтроллера в буфер.

Program : Программирование или загрузка кода HEX в буфер программной памяти целевого микроконтроллера.

Verify : Проверка показного программирования, сравнение с буфером.

Settings

Communication : Выбор коммуникационного порта между ПК и целью микроконтроллера.

Command Window : Выбор окна для отображения результата.

Help : Просмотр всей инструкции и пользование обратным программным обеспечением.

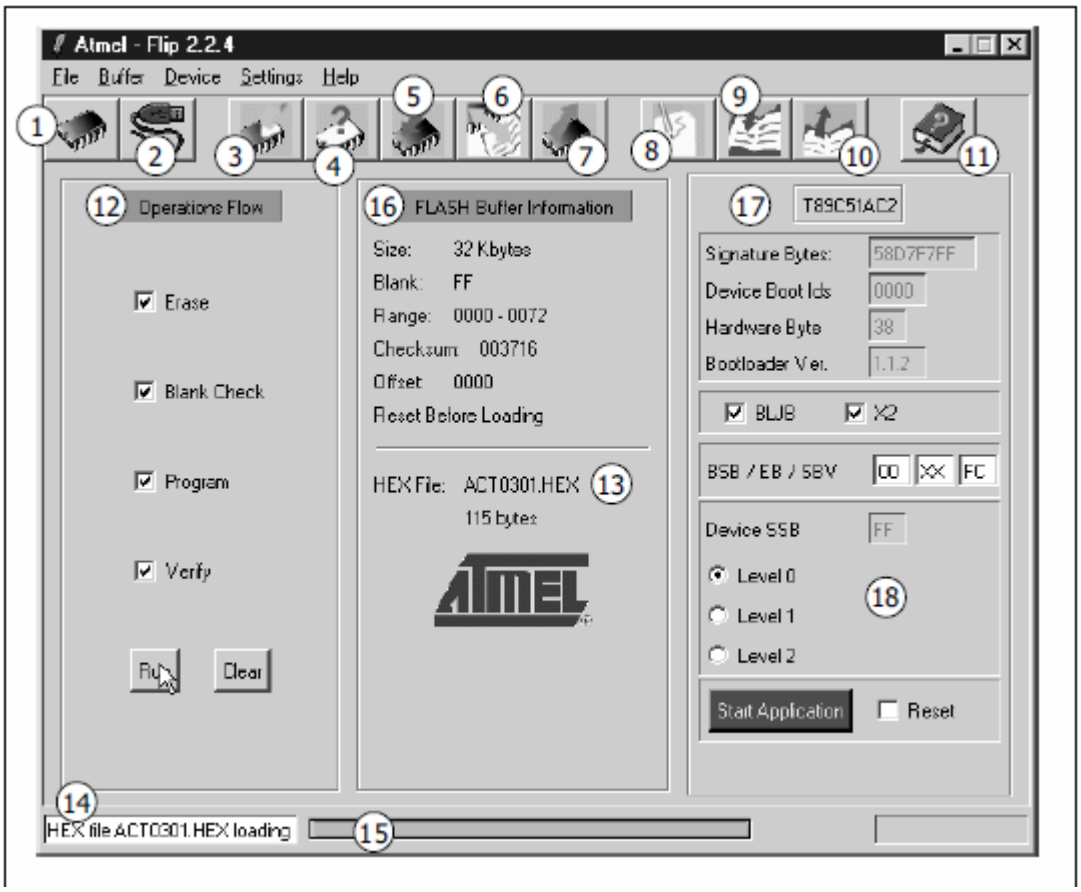


Рисунок 2.17 – Детали в FLIP

1. Выберите целевое устройство или кнопку микроконтроллер.
2. Установите коммуникацию или компьютерную портовую кнопку, которая соединяется с целью микроконтроллера.
3. Сотрите программную флеш-память в пределах целевого устройства.
4. Кнопка проверки заполнения.
5. Кнопка программирования целевого устройства.
6. Кнопка проверки устройства.
7. Кнопка чтения устройства; читайте данные из целевого микроконтроллера, чтобы показать буфер.
8. Кнопка буферного редактора; определите, что значение изменяется в буфере.
9. Загрузите файл; импортируйте подготовленный файл после компилирования или сборки в буфере.
10. Кнопка сохранения файла прошивки.

11. Кнопка помощи, можно увидеть некоторую информацию по программному обеспечению.
12. Командный список для автоматического программирования в потоке операций. Выполняется функция, которая проверяет проект перед этой процедурой, как указано ниже: Erase/Blank check/Program/Verify. После урегулирования щелкните кнопку RUN для последующего действия. Щелкните кнопку Clear, если не требуется автопрограммирование.
13. Окно показа выбора файла HEX.
14. Окно статуса.
15. Окно выполнения.
16. Выберите и установите буферную информацию.
17. Суммарная информация целевого микроконтроллера включает:
    17.1 Информационное число целевого микроконтроллера.
    17.2 Байт подписи.
    17.3 Устройство Boot IDS.
    17.4 Аппаратный байт.
    17.5 Загрузочную версию.
    17.6 BLJB(Bootloader Jump Bit) - этот параметр должен выбрать разрешение системного программирования.
    17.7 X2: этот параметр – временное действие. Если выбрать, то цикл будет действовать 6 часов. Но если не выбирать, то будет 12-часовое действие. В наборе Robo-51 нужно выбрать этот параметр.
    17.8 Уровни 0-2 Кодовый защитный уровень.

Как загрузить файл к микроконтроллеру с Flip

1) Соедините RBX-51AC2 плату управления с портом ПК RS232.
2) Включите электроснабжение. Выключатель на печатной плате для выбора программного метода. Светодиод LED PGM засветиться и тогда снова устанавливают целевой микроконтроллер. Предположительно устройство будет содержать запрограммированную ранее программу FLIP. Аппаратный ISP будет описан далее по этому руководству.
3) Установите FLIP.
4) Выберите устройство из списка :
    4.1) От строки высшего меню, выполните Device→Select. Диалоговое окно выбора появится само. Выберите контролер из списка и щелкните ОК.
    4.2) Как только контролер выбран, окно параметров устройства обновится, и вы сможете увидеть свободные специально предложенные байты устройства. Окно буферной информации также обновится с информацией об устройстве.

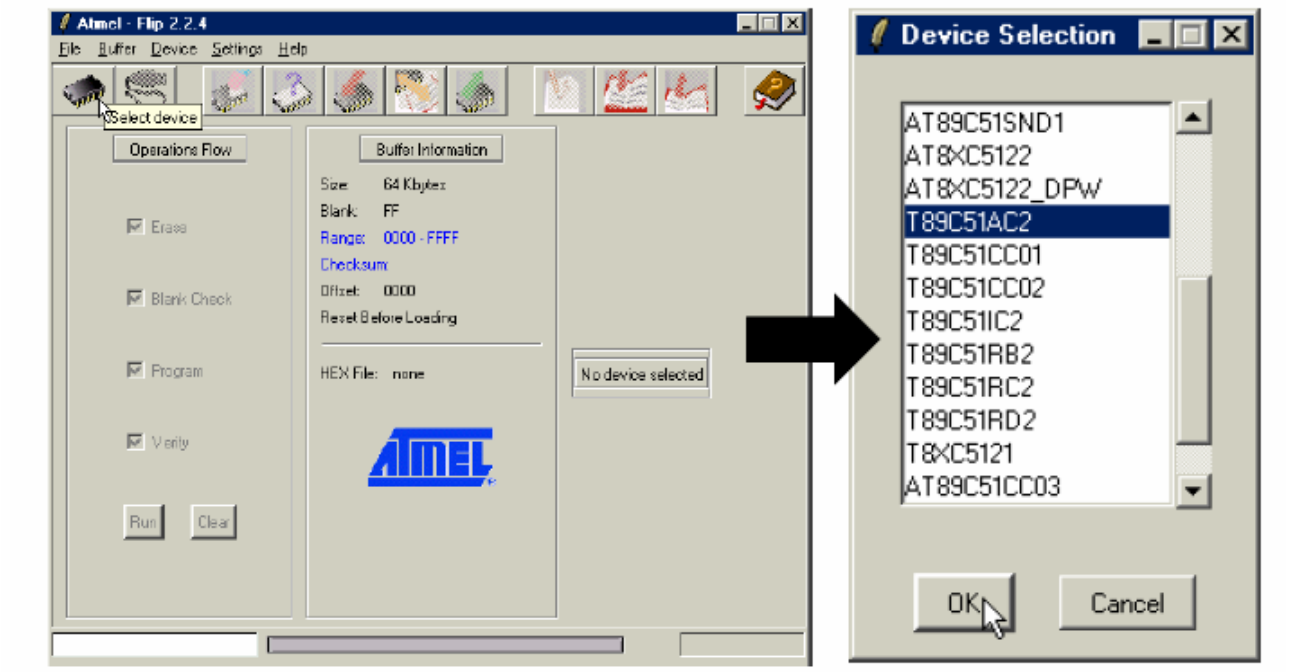


Рисунок 2.18 – Выбор процессора в Flip

5) Выберите средство коммуникации:

5.1) В строке высшего меню Settings→Communication выберите RS- 232.

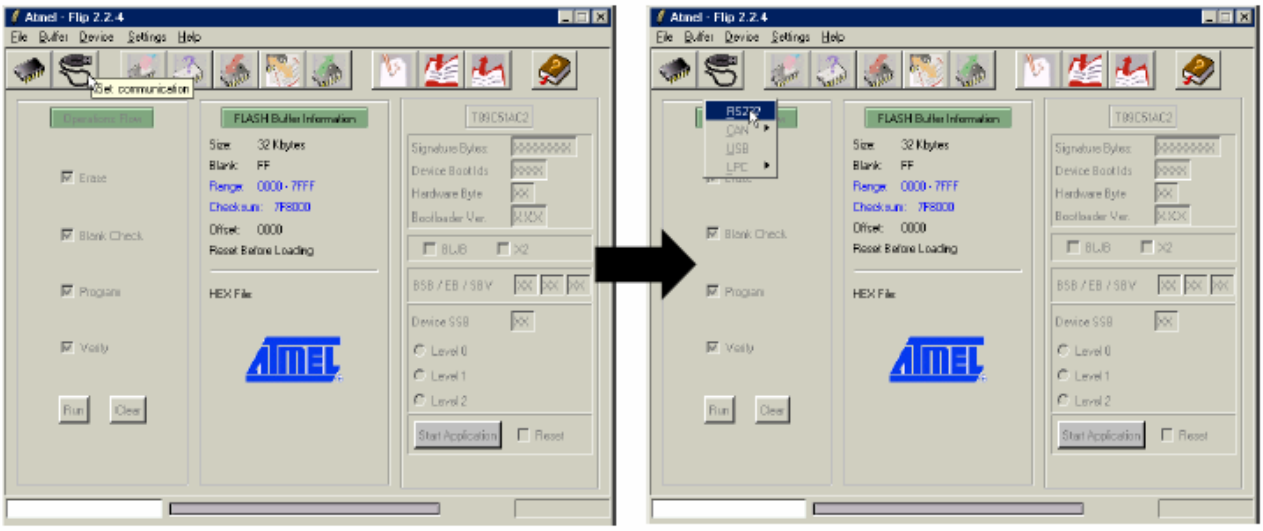


Рисунок 2.19 – Выбор порта коммуникации в Flip

5.2) RS-232 выше установочное диалоговое окно. Регулируйте (при необходимости) параметры коммуникации и щелкните Connect.

Flip запускает в ход последовательность синхронизации с целевым устройством загрузки проекта. После завершения последовательности синхронизации Flip читает целевые байты в предложенном устройстве и обновляет главное окно справа.

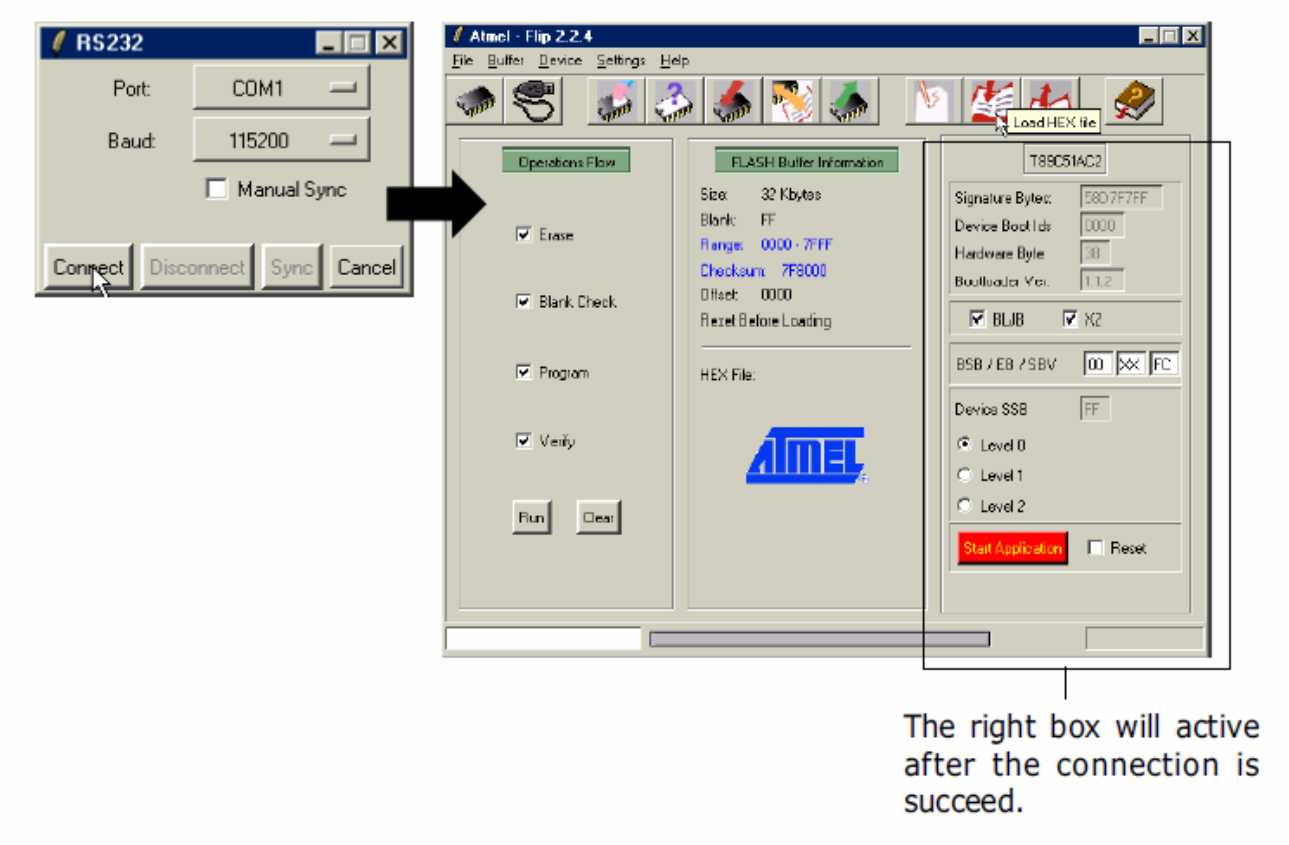


Рисунок 2.20 – Последовательность синхронизации

6) Выбирают HEX файл данных:

6.1) В строке высшего меню, выполняют File →LoadHEX File... или щелкают кнопку  .

6.2) Выберите файл HEX в файловом браузере.

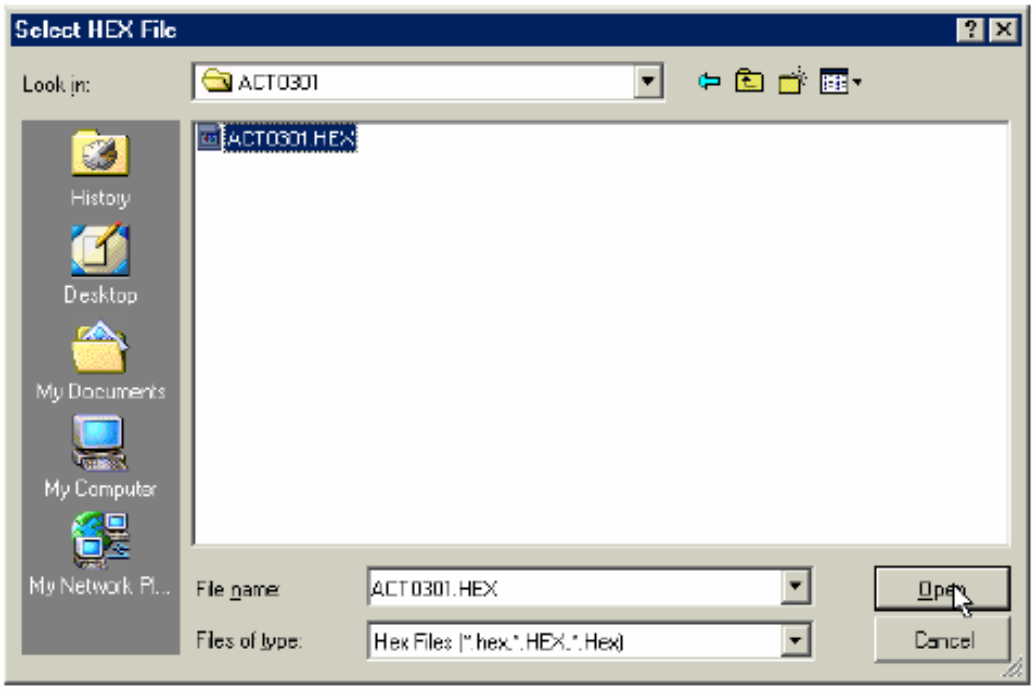


Рисунок 2.21 – Выбор HEX файл данных

7) В коробке операционного потока в левой стороне щелкните инструкции в заказе от вершины, чтобы сделать как указано ниже Erase→Blank→Check→Program→Verify. После этого щелкните кнопку RUN.

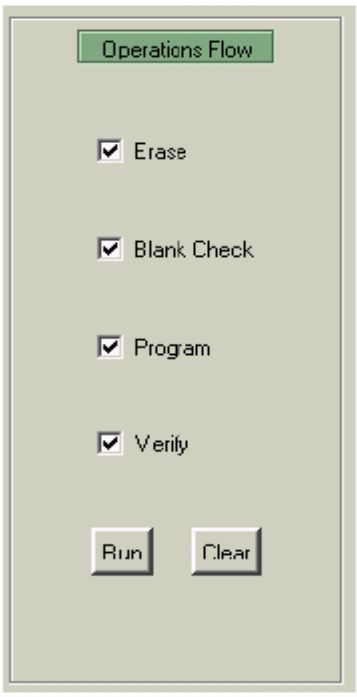


Рисунок 2.22 – Фигура 6 представляющая загрузку файла с Flip

FLIP сотрет все данные в программной флеш-памяти, проверит пустую, напишет HEX код с флеш и EEPROM памяти и проверит?. Все

операции выполняются автоматически. Пользователь может привести часть работы.

Дополнение к инструкции

Прибавление к флеш-программированию, FLIP имеют больше инструкций, подобных профессиональному программному обеспечению, которые указаны ниже :

(8) Откройте окно буферного издания:

(8.1) От строки высшего меню выполняют Buffer→Edit.

(8.2) Окно буфера редактирования возникает неожиданно. Сейчас возможно выполняете многих действия в буферных объемах.

(9) Откройте окно буферного выбора:

(9.1) Вы, возможно, открываете окно буферного выбора от FLIP окна магистрали или от диалогового окна буфера редактирования.

(9.2) От строки меню главного окна выполняют Buffer→Options. Диалоговое окно выбора возникает выше. Имеется главный буферный выбор:

- буферный размер;
- начальные объемы;
- адресная амплитуда программирования;

(10) Проверьте целевое устройство :

От строки высшего меню, выполняют Device→Verify.

Для детальных описаний возможных действий посмотрите Operations Summary в файле Help программного обеспечения.

Задание по выполнению лабораторной работы:

1. Изучение процессора на базе микросхемы T89C51AC2.
2. Изучение платы управления и её основных компонентов на базе микросхемы T89C51AC2.
3. Написание тестовой программы для платы управления мехатронной системой IE-ROBO-51.
4. Запуск и проверка тестовой программы на лабораторном стенде IE-ROBO-51.

Контрольные вопросы:

1. Структура и принципы построения мехатронных систем.
2. Способы управления мехатронными системами.
3. Интеллектуальные мехатронные системы.

Лабораторная работа №3

Программное управление визуализацией и звуковым сопровождением МС

Цель работы:

Изучить основные шаги для программного управления визуализацией, звуковым сопровождением МС и ознакомиться с LCD интерфейсом на плате RBX-51AC2.

Основные теоретические сведения:

Управление светодиодом на плате RBX-51AC2

На схеме (рис. 3.1) светодиод будет связано в раковине текущего типа. Резистор на 1 кВт ограничивает ток резистора для светодиода. После сброса по включении питания все порты начальной логики "HIGH" все светодиоды будут выключены. Таким образом светодиод должен отправить логику "LOW" для каждого порта.

В программировании необходимо сделать программу для отправки логики "LOW" в порт и задержку для поддержания логики "LOW" для человеческого видения. Если бы не задержка, то человек не мог бы видеть светодиодные операции, так как микроконтроллер работает очень быстро. Задержка – это очень важно. Однако в инструменты программирования C не обеспечивает функцию задержки. Пользователи должны сделать собственную??? и сохранить как файл библиотеки.

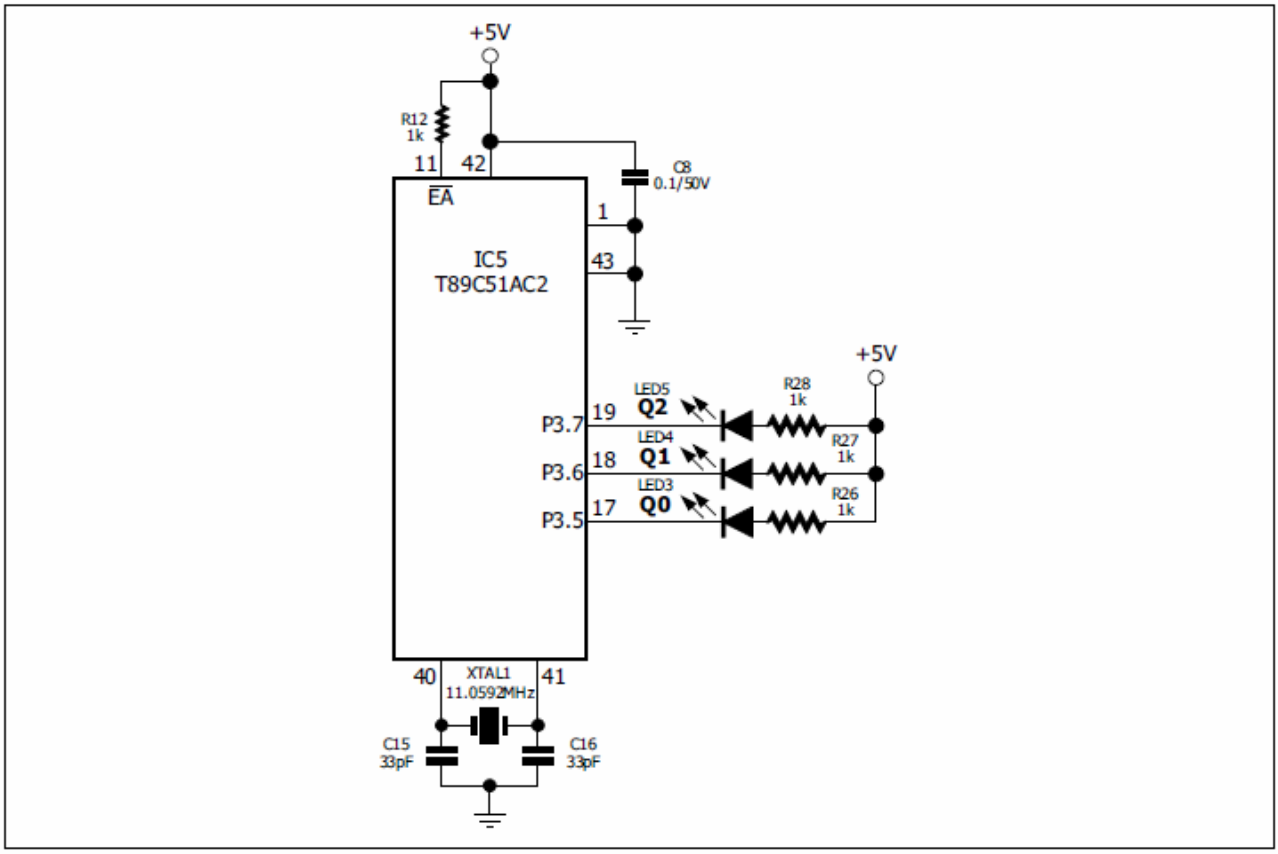


Рисунок 3.1 – Схема управления на плате RBX-51AC2

На рис. 3.2 представлен исходный код библиотеки, которая является функцией временной задержки в 100 микросекунд и 1 миллисекунду.

```
/*****************************************************************************/
// Program     : Delay 100us and 1 ms per unit
// Filename    : delay_robo51.h
// C compiler : RIDE 51 V6.1
/*****************************************************************************/
/*************************** Delay 1 ms per unit ***************************/
void delay_ms(unsigned int ms)
{
  unsigned int x,a;           // Keep for counter loop
  for(x=0;x<ms;x++)
  {
    for(a=0;a<878;a++);       // Loop for delay 1 ms per unit
  }
}
/************************ Delay 100 us per unit ******************************/
void delay_100us(unsigned int us)
{
  unsigned int x,a;           // Keep for counter loop
  for(x=0;x<us;x++)
  {
    for(a=0;a<75;a++);        // Loop for delay 100 us per unit
  }
}
```

Рисунок 3.2 - Исходный код библиотеки delay_robo51.h

Библиотеки Delay_robo51

Рисунок 3.2 является исходным кодом delay_robo51.h. Файл задержки библиотеки, которые используют во всех экспериментах в комплекта Robo-51. Пользователь может сделать с типом программы в блокноте или любом редакторе и сохранить в h file. Подробная история на c:\ride\inc.

Описание функции заключается в следующем:

(1) delay_ms: Задержка в миллисекундах

Функция формата:

void delay_ms (неподписанных Int мс)

Параметр:

ms - время задержки в миллисекундах. Диапазон от 0 до 65535

(2) delay_100us: Задержка в 100 мкс или 0,1 миллисекунд

Функция формата:

void delay_100us (неподписанных Int нас)

Параметр:

us - время задержки устройства в 100 микросекунд. Диапазон от 0 до 65535

Пример A3-1

delay_100us (5); / / Задержка 500 микросекунды

delay_100us (20); / / Задержка 2000 микросекунд или 2 миллисекунды

delay_ms (2000); / / Задержка 2 минуты

Открытие RIDE и создание нового проекта

- Сделайте C код (см. рис. 3.3) и сохранить как act01.c файл. Сборка проекта в HEX файл.

- Применить напряжение питания плате RBX-51AC2 контроллера.

- Скачать шестнадцатеричный файл с шагом A1.1 с FLIP программой и запустить. Соблюдайте операции.

3-точки Запуск светодиодная будет работать. Она начинается с P3.5 P3.7 в. Индикатор будет с 2-секунд.

На рис. 3.3 представлена программа на C++ для запуска диода на плате RBX-51AC2.

```
/*---------------------------------------------------------------------------*/
// Program     : LED basic drive
// Description: drive LED P3.5 , P3.6 and P3.7 by delay interval 2 seconds
// Filename    : act0301.c
// C compiler : RIDE 51 V6.1
/*---------------------------------------------------------------------------*/
#include <C51ac2.h>          // Declare T89C51AC2's register
#include <delay.h>           // Declare Delay library
sbit led1 = P3^5;            // Define led1 to P3.5
sbit led2 = P3^6;            // Define led2 to P3.6
sbit led3 = P3^7;            // Define led3 to P3.7
void main()
{
  while(1)                   // Endless loop
  {
    led1 = 0;                // LED at P3.5 on
    delay_ms(2000);          // Delay 2 seconds from delay function of
                             // "delay.h" library
    led1 = 1;                // Turn-off LED at P3.5

    led2 = 0;                // Turn-on LED at P3.6
    delay_100us(20000);      // Delay 2 seconds from delay_100us function
                             // from Delay library
    led2 = 1;                // Turn-off LED at  P3.6

    led3 = 0;                // Turn-on LED at  P3.7
    delay_100us(20000);      // Delay 2 seconds from delay_100us function
                             // from Delay library
    led3 = 1;                // Turn-off LED at P3.7
  }
}
```

Рисунок 3.3 - Программа запуска диода на плате RBX-51AC2.

LCD интерфейс в плате RBX-51AC2

Схема интерфейса показана на рисунке 3.4. Шестиконтактный порт включает P0.4, P0.7 с контактами для данных D4 по D7 в режиме с 4-битным интерфейсом, P0.2 контактный с RS и P0.3 в контакте с импульсом E.

R/W контакт LCD соединен с землей только для записи всех данных на LCD.

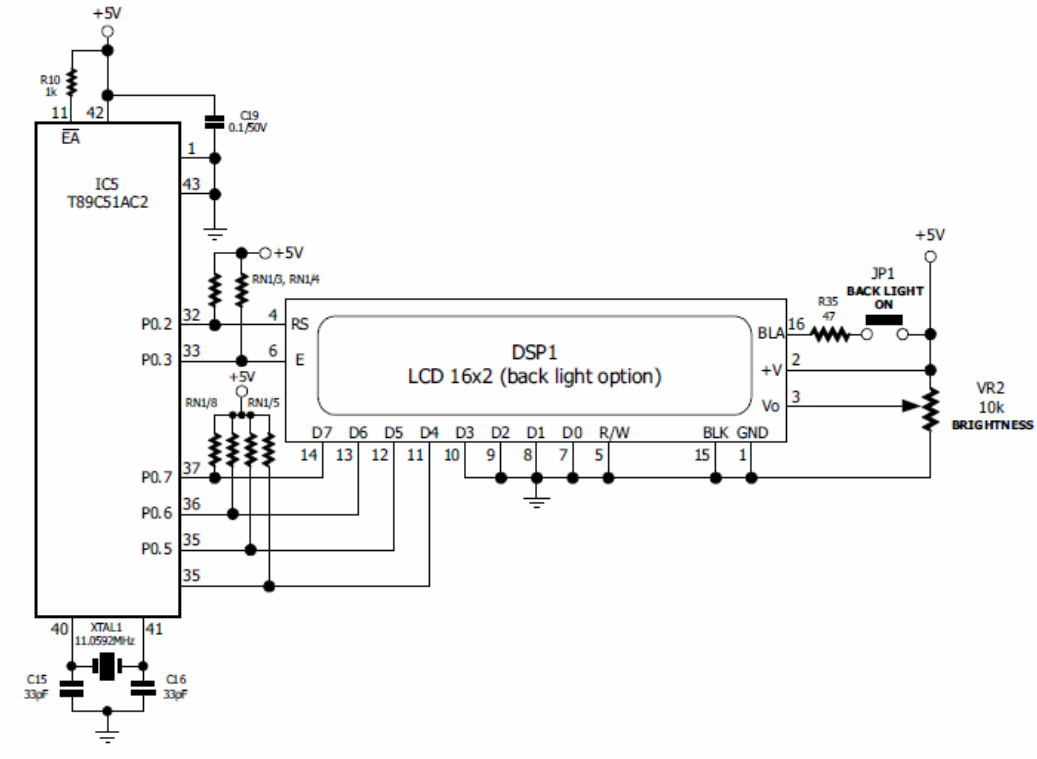


Рисунок 3.4 – Схема LCD интерфейса

Набор инструкций LCD-модуля

1. Очистить экран

Инструкция данные л/р 1.

2. Возвращение домой

Данные инструкции в л/р 2 .

Переместить курсор в исходное положение, в результате отображение не изменится.

3. Набор режим ввода

Формат инструкции данных представлен на рисунке 3.5.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0	1	I/D	S

Рисунок 3.5 – Формат инструкции 1

I / D бит: установить DDRAM адрес после записи или чтения данных;

"0" - уменьшить адрес;

"1" - увеличение адреса;

S бит: установить режим ввода;

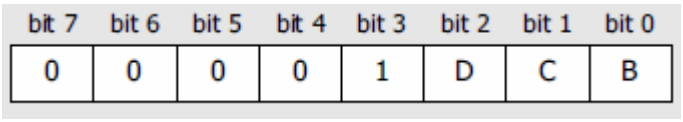
"0" - автоматическая смена курсора сразу после байта;

"1" - курсор не двигается. Новый ход вместо символа слева.

Это значит, сдвиг курсора вправо, когда новый символ entried и DDRAM адрес увеличится.

4. Управление дисплеем

Формат инструкции данных представлен на рисунке 3.6.



bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	0	1	D	C	B

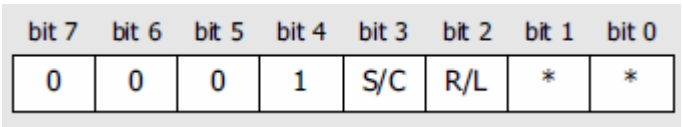
Рисунок 3.6 - Формат инструкции 2

D бит: ON / OFF экран;
"0" – выкл. LCD-экран;
"1" – вкл. LCD –экран;
C бит: ON / OFF курсор;
"0" – выкл. курсор;
"1" – вкл. курсор;
B бит: мигающий курсор контроль;
"0" - курсор не мигает;
"1" - курсор мигает.

Инструкция данных составляет $ 0C. Это означает, что на LCD - экран, происходит выключение курсора. Другой $ 0F. Это означает, что на LCD-экран, происходит включение и курсор мигает.

5. Переход контроля

Формат инструкции данных представлен на рисунке 3.7.



bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	0	1	S/C	R/L	*	*

Рисунок 3.7 - Формат инструкции 3

Таблица 3.1 – Формат

S / C	R / L	Переключение формата	Данные
0	0	Сдвиг курсора влево	$ 10 - $ 13
0	1	Сдвиг курсора вправо	$ 14 - $ 17
1	0	Сдвиг новый символ. влево	$ 18 - $ 1B
1	1	Сдвиг новый символ. вправо	$ 1C-$ 1F

6. Функция управления
Формат инструкции данных представлен на рисунке 3.8

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	0	1	DL	N	F	*	*

Рисунок 3.8 - Формат инструкции 4

DL бит: установите режим интерфейса.4
"0" - 4-битный режим;
"1" - 8-битном режиме;
N бит: Выберите строке дисплея;
"0" - 1-LINE;
"1" - 2-линий или более;
F бит: выберите разрешение характера;
"0" - дисплей 5x7 пунктов;
"1" - дисплей 5x10 пунктов;

Передача звука

Микроконтроллер создает звуковые сигналы, быстро посылая высокий/низкий сигналы к различным типам динамиков. Скорость этих высокого/низкого сигналов, называется частотой, и это определяет тона и шаг звукового сигнала. Если каждый раз высокий/низкий сигналы повторяются, то это цикл. Вы часто будете видеть число циклов в секунду, которые называются герцами - сокращенно Гц. Например, одним из наиболее распространенных частоты для сигнала, что машине помочь привлечь ваше внимание составляет 2 кГц. Это означает, что высокий/низкий сигналы повторяются 2000 раз в секунду.

Представляем пьезоэлектрический динамик

В этом упражнении вы будете экспериментировать с передачей сигналов в общем, малых и недорогой говорящим названием пьезоэлектрический динамик. Как работает схема пьезоэлектрического динамика? Когда гитарная струна вибрирует, это вызывает изменения в давлении воздуха. Эти изменения в воздухе - давление, т.е. то, что ухо распознает как тон. Чем быстрее изменения в давлении воздуха, тем выше высота, тем медленнее изменения в давлении воздуха, тем ниже поле. Элемент внутри пластикового корпуса пьезодинамик, называется пьезоэлектрическим элементом. Когда высокий/ низкий сигналы применяются к положительным выводам докладчика, пьезоэлектрический элемент вибрирует, и это вызывает изменения в давлении воздуха так же,

как гитарная струна: ухо обнаруживает изменения в давлении воздуха, вызванное пьезоэлектрическим динамиком, и это обычно звучит как звуковой сигнал или тон.

Передача звукового сигнала на плату контроллера RBX-51AC2

На плате RBX-51AC2 контроллера имеется динамик Piezo. Это генерация сигнала через P2.7 из T89C51AC2. Сигнальные входы в простую схему транзисторного усилителя и управляющий сигнал на динамик пьезодинамик. Схема интерфейса показана на рисунке 3.9.

В программировании микроконтроллеров необходимо сделать вверх/вниз сигнал на порт P2.7. Простой прямоугольный сигнал показан на рисунке 3.10. Если вы сделаете этот сигнал продолжительным, то сможете определить период генерирующий звук.

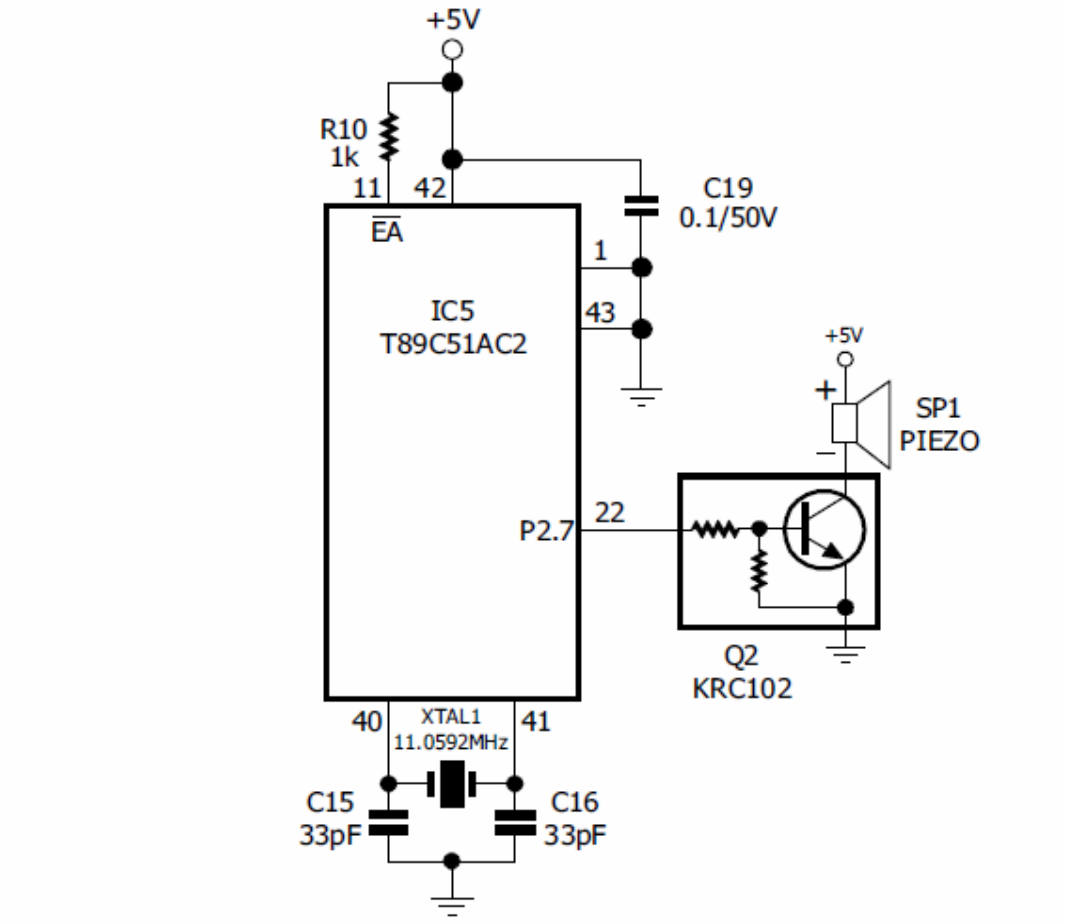


Рисунок 3.9 – Схема транзисторного усилителя

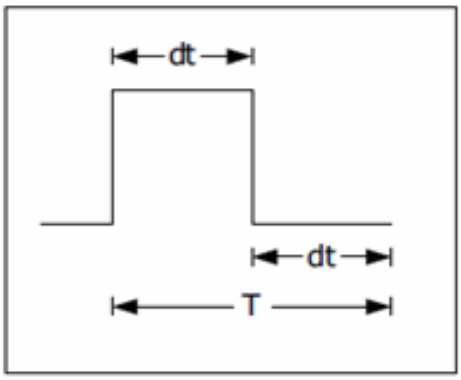


Рисунок 3.10 – График прямоугольного сигнала

Сигнал имеет цикл в 100 мкс. Вы можете рассчитать частоту следующим образом:

$$f=\frac{1}{T}=\frac{1}{2*100*10^{-6}*dt}=\frac{5000}{dt}. \quad (3.1)$$

Вы можете определить длительность звукового сигнала программно с набранным количеством периодов. На рисунке 3.11 генерируется сигнал продолжительностью со временем параметра в миллисекундах. Вы можете найти число периодов по формуле:

$$\text{Количество периодов или цикл} = time*\frac{10^{-3}}{2*100*10^{-6}*dt}=\frac{5*time}{dt}. \quad (3.2)$$

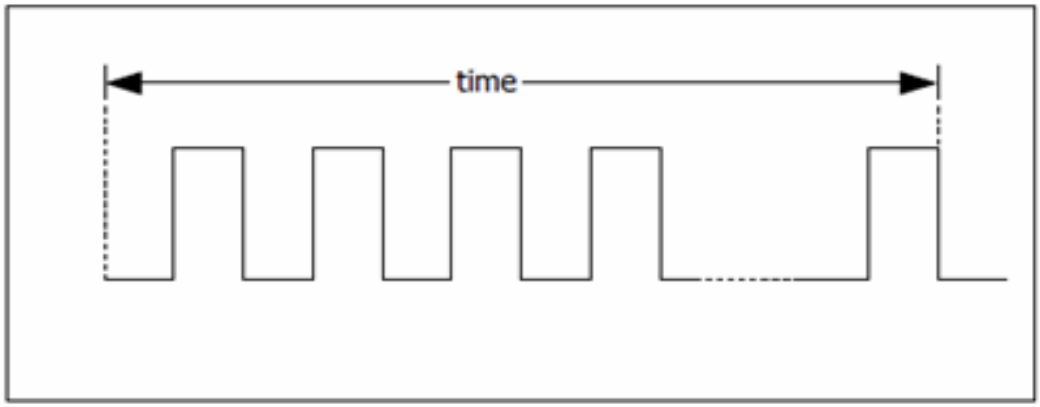


Рисунок 3.12 – График продолжительности прямоугольного сигнала

Звуковая библиотека

Программа в приложении А является исходным кодом sound_robo51.h.Файл задержки библиотеки, который используют во всех экспериментах комплекта Robo-51. Пользователь может сделать типом программы в блокноте или любом редакторе и сохранить в .h file.

Описание функции заключаются в следующем:

(1) delay_sound: установить время задержки для генерации звука сигнала

Формат функции:

void delay_sound (неподписанных Int мс)

Параметр:

Ms - время задержки в миллисекундах. Диапазон от 0 до 65535.

(2) звук: генерация звукового сигнала

Формат функции:

void delay_sound (Int частота, Int время)

Параметр:

freq - Целевая частота в Гц;

time - продолжительность времени во второй блок.

(3) звуковой сигнал: генерация звукового сигнала на частоте 500 Гц 0,1 секунд.

Формат функции:

void delay_sound (недействительными)

Пример:

sound (700,200); / / Создание сигнала частотой 700Hz с длительностью 200 миллисекунд

beep (); / / Создание звукового сигнала

- Открыть RIDE и создать новый проект. Сделайте C код в программе (Приложение Б) и сохраните как act.c файл. Сборка проекта в HEX файл.

- Скачать шестнадцатеричный файл и запустить FLIP программу. Соблюдайте операции.

- Светодиод на P3.5 P3.7 к свету и можно слушать звук разной частоты.

Порт P3.5 с частотой 1 кГц, P3.6 - 800Hz и 500Hz для P3.7.

Приложение А

```
/*---------------------------------------------------------------------------*/
// Program : Generate sound by frequency
// Description: Generate sound function
// Filename : sound_robo51.h
// C compiler : RIDE 51 V6.1
/*---------------------------------------------------------------------------*/
sbit s_bit = P2^7;
#pragma DISABLE // for disable all interrupt
// before call function delay_sound
void delay_sound(unsigned int ms)
{
unsigned int x,a; // Keep for counter loop
for(x=0;x<ms;x++)
{
for(a=0;a<75;a++); // Loop for delay 100 us per unit
}
}
#pragma DISABLE // Disable all interrupt before call ‘sound’ function
void sound(int freq,int time)
{
int dt=0,m=0; // Keep value and keep active logic delay time
dt = 5000/freq;
time = (5*time)/dt; // Keep counter for generating sound
for(m=0;m<time;m++)
{
s_bit = 1; // P2.7 = high
delay_sound(dt); // Delay for sound
s_bit = 0; // P2.7 = low
delay_sound(dt); // Delay for sound
}
}
#pragma DISABLE // Disable all interrupt before call ‘beep’ function
void beep(void)
{
sound(500,100); // Generate sound at default frequency
}
```

Note :
In using all function of this library do not disturb from any interrupt. Because put #pragma DISABLE directive before call each function.

Приложение Б

```
/*---------------------------------------------------------------------------*/
// Program : LED and sound
// Description: drive LED P3.5, P3.6 and P3.7 by delay interval 2 seconds
// and generate sound when LED changing
// Filename : act0401.c
// C compiler : RIDE 51 V6.1
/*---------------------------------------------------------------------------*/
#include <C51ac2.h> // Declare T89C51AC2's register
#include <delay_robo51.h> // Declare Delay library
#include <sound_robo51.h> // Declare Sound generator library
sbit led1 = P3^5; // Define led1 to P3.5
sbit led2 = P3^6; // Define led2 to P3.6
sbit led3 = P3^7; // Define led3 to P3.7
void main()
{
while(1) // Endless loop
{
sound(1000,400); // Generate sound at 1000Hz for 0.4 second
led1 = 0; // Turn-on LED at P3.5
delay_ms(2000); // Delay 2 seconds
// (call function from Delay library)
led1 = 1; // Turn-off LED at P3.5
sound(800,200); // Generate sound at 800Hz for 0.2 second
led2 = 0; // Turn-on LED at P3.6
delay_100us(20000); // Delay 2 seconds
// (call function from Delay library)
led2 = 1; // Turn-off LED at P3.6
beep(); // Generate "beep" signal at 500Hz for 0.1s
led3 = 0; // Turn-on LED at P3.7
delay_100us(20000); // Delay 2 seconds
led3 = 1; // Turn-off LED at P3.7
}
}
```

Задание по выполнению лабораторной работы:

1. Изучить управление светодиодами.
2. Написать и запустить тестовую программу по светодиодной визуализации.
3. Изучить LCD интерфейс.
4. Написать и запустить тестовую программу по LCD визуализации.
5. Изучить звуковое сопровождение МС.
6. Написать и запустить тестовую программу по звуковой визуализации.

Контрольные вопросы:

1. Сферы применения мехатронных систем.
2. Способы вывода информации.
3. Построение модулей вывода информации.

Лабораторная работа №4

Программное управление простейшими движениями МС IE-ROBO-51

Цель работы:

Изучить принципы управления движениями мехатронной системы IE-ROBO-51, написать программу для тестирования простейших движений МС IE-ROBO-51 .

Основные теоретические сведения:

Robo-51имеет 2 канала DC драйверов двигателя. Вы можете контролировать скорость и направление вращения двигателя постоянного тока с программным обеспечением. Из-за постоянного тока приводится в движение PWM (Пульс импульсной модуляции) сигнала. В этом разделе описывается управление двигателем постоянного тока с PWM, схемой цепи для генерации PWM сигнала T89C51AC2 микроконтроллера и связанных с библиотекой файлов программирования C.

Основные операции для двигателя постоянного тока с PWM

При изменении (модуляции) ширина импульса, подаваемого на двигатель постоянного тока, может увеличить или уменьшить количество энергии, предоставляемого двигателю, тем самым увеличивая или уменьшая скорость двигателя. Обратите внимание, что, хотя напряжение имеет фиксированную амплитуду, оно имеет переменную цикла. Это означает, что чем шире импульс, тем выше скорость (см. рис. 4.1).

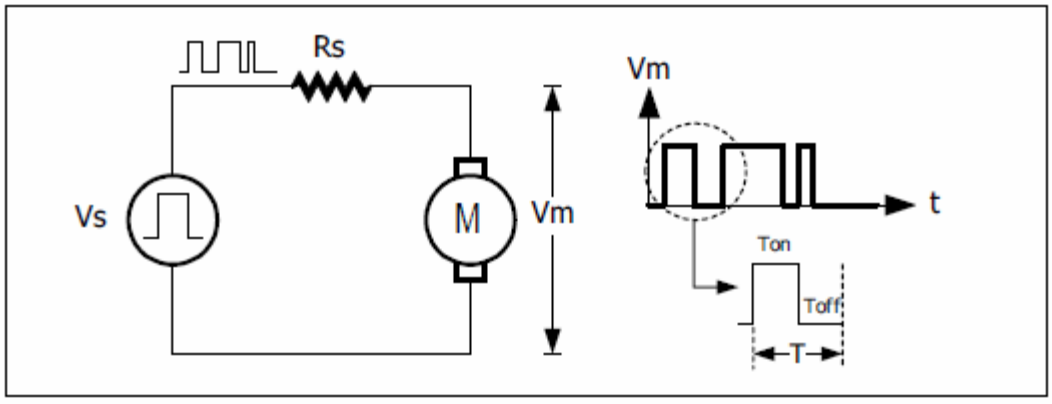


Рисунок 4.1

Хотя рабочий цикл определяет скорость двигателя, двигатель постоянного тока может работать на определенной частоте. Если частота PWM превышает лимит, DC двигатель остановится, потому что доберется до точки насыщения. На рисунке 3.9 показана работа двигателя постоянного тока с циклом PWM.

Например, PWM-сигнал на рисунке 4.1 имеет период 20 миллисекунд и частоту 50 Гц.

На рисунке 4.2 (A) цикл PWM составляет 20%. Двигатель будет вращаться с минимальной скоростью, потому что падение напряжения только 0.9 В. Когда увеличится рабочий цикл на рисунке 4.2 (B) и (C), напряжение, подаваемое на двигатель постоянного тока, увеличится. Его скорость увеличится тоже.

На рисунке 4.2 (D) напряжение, подаваемое на двигатель постоянного тока, высокого уровня, потому что рабочий цикл составляет 100%. Таким образом, контроль заполнения PWM является методом управления скоростью двигателя.

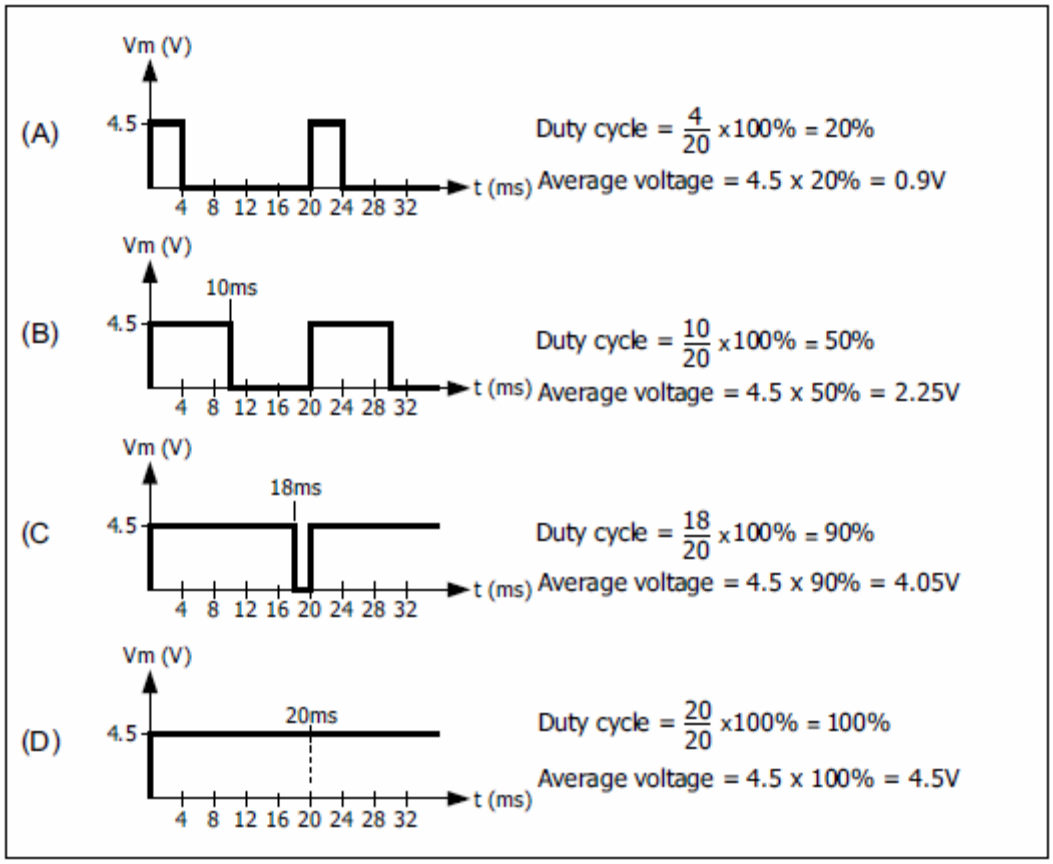


Рисунок 4.2 – Циклы с различными PWM

Назначение контактов

См. драйвер двигателя постоянного тока схемы Robo-51 (рис. 4.3), порт контактов T89C51AC2, связанных с L293D h-мост-драйвер имеет 6 контактов следующим образом :

P2.0 и P2.1are подключаются с IN1A и IN2A из L293D для управления мощностью двигателя на выходе 1 (1Y и 2Y контакт).

P1.6 связан с 12EN, чтобы включить контакт L293D. P1.6 контакт отправляет PWM сигнал для управления скоростью двигателя на выходе A на 1Y и 2Y контакта L293D.

P2.2 и P2.3, связаны с IN4A и IN3A (L293D) для управления двигателем на выходе 2 (3Y и 4Y-контакт).

P1.7 связано с 34EN, чтобы включить контакт с L293D. P1.7 контакт отправляет PWM сигнал для управления скоростью двигателя на выходе B на 3Y и 4Y контакт L293D.

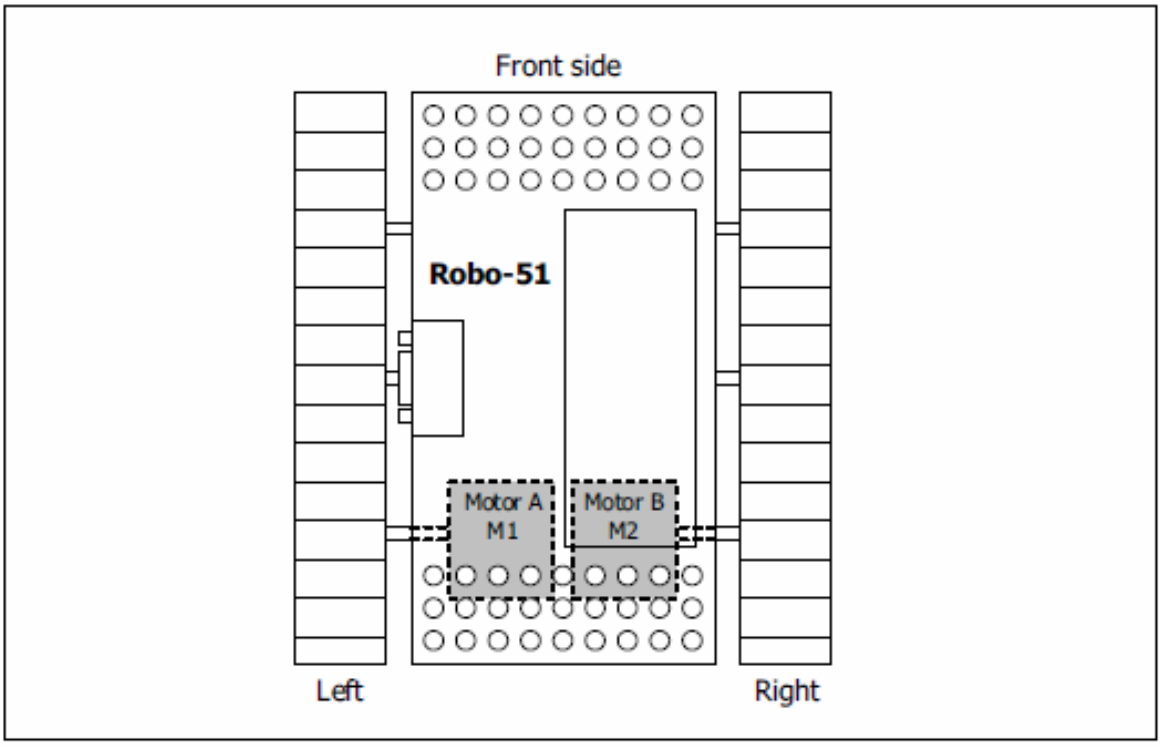


Рисунок 4.3 – Вид сверху на лабораторный макет

При создании двигательных функций Robo-51 должно учитываться:

1. Если смотреть на робота сверху, драйвер двигателя 1 мотора A означает, что это левая сторона двигателя и драйвер двигателя 2 мотора B означает, что это правая сторона двигателя.

2. Обратная сторона Robo-51 является стороной, куда крепятся редукторы двигателя постоянного тока.

3. Передняя сторона Robo-51 является стороной, куда крепятся 3-LED индикаторы.

Рисунок 4.3 показывает данные особенности.

Функция движения вперед

Для движения Robo-51 вперед необходимо запрограммировать Robo-51, в этом вам помогут функция motor_fd исходный код этой функции (см. рис. 4.4):

select_fd : select motor output channel
“1” - Motor output 1 (M-1)
“2” - Motor output 2 (M-2)
speed : determine the motor speed. Range is 0 to 255. At 255 is highest speed.

```
#define motor_pulse 115    // Define constant
sbit dir_a1 = P2^0;        // bit drive motor0
sbit dir_a2 = P2^1;        // bit drive motor0
sbit dir_b1 = P2^2;        // bit drive motor1
sbit dir_b2 = P2^3;        // bit drive motor1
void motor_fd(unsigned char select_fd, unsigned char speed)
{
   TMOD &= 0xF0;           // Refresh mode timer 0
   TMOD |= 0x02;           // Setup mode timer 0 (8 bit auto reload)
   TH0 = motor_pulse;      // Reload value for timer 0
   TL0 = motor_pulse;      // Initial value for count of timer 0
   TR0 = 1;                // Start timer 0
   CMOD = 0x04;            // Set CMOD PCA count freq. by pulse overflow timer0
   if(select_fd==1)
   {
     CCAPM3 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
     CCAP3L = 255-speed;   // Set CCAP3L initial value by speed variable
                           // (set duty cycle)
     CCAP3H = 255-speed;   // Set CCAP3H initial value by speed variable
                           // (set duty cycle)
     dir_a1 = 0;           // Set direction forward
     dir_a2 = 1;           // Set direction forward
   }
   else if(select_fd==2)
   {
     CCAPM4 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
     CCAP4L = 255-speed;   // Set CCAP4L initial value by speed variable
                           // (set duty cycle)
     CCAP4H = 255-speed;   // Set CCAP4H initial value by speed variable
                           // (set duty cycle)
     dir_b1 = 0;           // Set direction forward
     dir_b2 = 1;           // Set direction forward
   }
   CCON = 0x40;            // Set PCA counter run
}
```

Рисунок 4.4 – Код программы

Движение в обратном направлении

Для движения Robo-51 назад необходимо запрограммировать Robo-51 при помощи функции motor_bk, исходный код этой функции (см. рис. 4.5):

select_bk : select motor output channnel
“1” - Motor output 1 (M-1)
“2” - Motor output 2 (M-2)
speed : determine the motor speed. Range is 0 to 255. At 255 is highest speed.

```
#define motor_pulse 115    // Define constant
sbit dir_a1 = P2^0;        // bit drive motor0
sbit dir_a2 = P2^1;        // bit drive motor0
sbit dir_b1 = P2^2;        // bit drive motor1
sbit dir_b2 = P2^3;        // bit drive motor1

void motor_bk(unsigned char select_bk ,unsigned char speed)
{
   TMOD &= 0xF0;           // Refresh mode timer 0
   TMOD |= 0x02;           // Setup mode timer 0 (8 bit auto reload)
   TH0 = motor_pulse;      // Reload value for timer 0
   TL0 = motor_pulse;      // Initial value for count of timer 0
   TR0 = 1;                // Start timer 0
   CMOD = 0x04;            // Set CMOD PCA count freq. by pulse overflow timer0
   if(select_bk==1)
   {
     CCAPM3 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
     CCAP3L = 255-speed;   // Set CCAP3L initial value by speed variable
                           // (set duty cycle)
     CCAP3H = 255-speed;   // Set CCAP3H reload value by speed variable
                           // (set duty cycle)
     dir_a1 = 1;           // set direction backward
     dir_a2 = 0;           // set direction backward
   }
   else if(select_bk==2)
   {
     CCAPM4 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
     CCAP4L = 255-speed;   // Set CCAP4L initial value by speed variable
                           //(set duty cycle)
     CCAP4H = 255-speed;   // Set CCAP4H initial value by speed variable
                           // (set duty cycle)
     dir_b1 = 1;           // set direction backward
     dir_b2 = 0;           // set direction backward
   }
   CCON = 0x40;            // Set PCA counter run
}
```

Рисунок 4.5 – Код программы

Функция торможения

Для реализации функции торможения Robo-51, необходимо запрограммировать Robo-51 в этом вам помогут: функция motor_stop, исходный код этой функции (см. рис. 4.6):

select_stop : select motor output channel to brake
select_stop = 1 _ Brake only M-1 channel
select_stop = 2 _ Brake only M2 channel
select_stop = 3 _ Brake both channels

```
sbit dir_a1 = P2^0;       // bit drive motor0
sbit dir_a2 = P2^1;       // bit drive motor0
sbit dir_b1 = P2^2;       // bit drive motor1
sbit dir_b2 = P2^3;       // bit drive motor1
void motor_stop(char select_stop)
{
   if(select_stop==1)
   {
     dir_a1 = 0;       // Break motor channel 1
     dir_a2 = 0;       // Break motor channel 1
     CCAPM3 &= 0xFD;   // Stop generate PWM at pin P1.6
   }
   else if(select_stop==2)
   {
     dir_b1 = 0;       // break motor channel 2
     dir_b2 = 0;       // Break motor channel 2
     CCAPM4 &= 0xFD;   // Stop generate PWM at pin P1.7
   }
   else if(select_stop==3)
   {
     dir_a1 = 0;       // Break motor channel 1
     dir_a2 = 0;       // Break motor channel 1
     CCAPM3 &= 0xFD;   // Stop generate PWM at pin P1.6
     dir_b1 = 0;       // Break motor channel 2
     dir_b2 = 0;       // Break motor channel 2
     CCAPM4 &= 0xFD;   // Stop generate PWM at pin P1.7
   }
}
```

Рисунок 4.6 – Код программы

Библиотека для Robo-51

Вы можете комбинировать и создавать новые файлы для библиотеки - dc_motor.h. Вы можете сделать это с текстом программы: отредактируйте и сохраните в dc_motor.h файл в папку C: \ RIDE \ INC.

Описание функции этой библиотеки можно резюмировать следующим образом (см. рис. 4.7):

(1) **motor_bk** : Backward movement function
Function format :
void motor_bk(bit select_bk ,unsigned char speed)
Parameter :

select_bk - Select motor channel. Range is 1 and 2.
speed - Determine motor speed. Range is 0 to 255
(2) **motor_fd** : Forward movement function
Function format :
void motor_fd(bit select_fd, unsigned char speed)
Parameter :
select_fd - Select motor channel. Range is 1 and 2.
speed - Determine motor speed. Range is 0 to 255
(3) **motor_stop** : Brake motor function
Function format :
void motor_stop(char select_stop)
Parameter :
select_stop - Select motor channel to stop. Range is 1, 2 and 3
(3means select both channels)

```
/*-------------------------------------------*/
// Program     : Pulse width modulation by PCA module
// Description: Generate pulse width modulation signal on CEX3 and CEX4
//             : using PCA function
// Filename    : dc_motor.h
// C compilar  : RIDE 51 V6.1
/*-------------------------------------------*/
#define ch1 1                 // Define constant
#define ch2 2                 // Define constant
#define all 3                 // Define constant
#define motor_pulse 115       // Define constant
sbit dir_a1 = P2^0;           // bit drive motor0
sbit dir_a2 = P2^1;           // bit drive motor0
sbit dir_b1 = P2^2;           // bit drive motor1
sbit dir_b2 = P2^3;           // bit drive motor1

/********************** Drive motor backward ********************************/
void motor_bk(unsigned char select_bk ,unsigned char speed)
{
   TMOD &= 0xF0;              // Refresh mode timer 0
   TMOD |= 0x02;              // Setup mode timer 0 (8 bit auto reload)
   TH0 = motor_pulse;         // Reload value for timer 0
   TL0 = motor_pulse;         // Initial value for count of timer 0
   TR0 = 1;                   // Start timer 0
   CMOD = 0x04;               // Set CMOD PCA count freq. by pulse overflow timer0
   if(select_bk==1)
   {
     CCAPM3 = 0x42;           // Set CCAP Module 3 as 8 bit PWM
     CCAP3L = 255-speed;      // Set CCAP3L initial value by speed variable
                              // (set duty cycle)
     CCAP3H = 255-speed;      // Set CCAP3H reload value by speed variable
                              // (set duty cycle)
     dir_a1 = 1;              // set direction backward
     dir_a2 = 0;              // set direction backward
   }
   else if(select_bk==2)
   {
     CCAPM4 = 0x42;           // Set CCAP Module 3 as 8 bit PWM
     CCAP4L = 255-speed;      // Set CCAP4L initial value by speed variable
                              // (set duty cycle)
     CCAP4H = 255-speed;      // Set CCAP4H initial value by speed variable
                              // (set duty cycle)
     dir_b1 = 1;              // set direction backward
     dir_b2 = 0;              // set direction backward
   }
   CCON = 0x40;               // Set PCA counter run
}

/********************** Drive motor forward ********************************/
void motor_fd(unsigned char select_fd, unsigned char speed)
{
   TMOD &= 0xF0;              // Refresh mode timer 0
   TMOD |= 0x02;              // Setup mode timer 0 (8 bit auto reload)
   TH0 = motor_pulse;         // Reload value for timer 0
   TL0 = motor_pulse;         // Initial value for count of timer 0
```

```
    TR0 = 1;                // Start timer 0
    CMOD = 0x04;            // Set CMOD PCA count freq. by pulse overflow timer0
    if(select_fd==1)
    {
      CCAPM3 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
      CCAP3L = 255-speed;   // Set CCAP3L initial value by speed variable
                            // (set duty cycle)
      CCAP3H = 255-speed;   // Set CCAP3H initial value by speed variable
                            // (set duty cycle)
      dir_a1 = 0;           // Set direction forward
      dir_a2 = 1;           // Set direction forward
    }
    else if(select_fd==2)
    {
      CCAPM4 = 0x42;        // Set CCAP Module 3 as 8 bit PWM
      CCAP4L = 255-speed;   // Set CCAP4L initial value by speed variable
                            // (set duty cycle)
      CCAP4H = 255-speed;   // Set CCAP4H initial value by speed variable
                            // (set duty cycle)
      dir_b1 = 0;           // Set direction forward
      dir_b2 = 1;           // Set direction forward
    }
    CCON = 0x40;            // Set PCA counter run
}

/*********************** Break motor ****************************************/
void motor_stop(char select_stop)
{
    if(select_stop==1)
    {
      dir_a1 = 0;           // Break motor channal 1
      dir_a2 = 0;           // Break motor channal 1
      CCAPM3 &= 0xFD;       // Stop ganarate PWM at pin P1.6
    }
    else if(select_stop==2)
    {
      dir_b1 = 0;           // break motor channal 2
      dir_b2 = 0;           // Break motor channal 2
      CCAPM4 &= 0xFD;       // Stop ganarate PWM at pin P1.7
    }
    else if(select_stop==3)
    {
      dir_a1 = 0;           // Break motor channal 1
      dir_a2 = 0;           // Break motor channal 1
      CCAPM3 &= 0xFD;       // Stop ganarate PWM at pin P1.6
      dir_b1 = 0;           // Break motor channal 2
      dir_b2 = 0;           // Break motor channal 2
      CCAPM4 &= 0xFD;       // Stop ganarate PWM at pin P1.7
    }
}
```

Рисунок 4.7 – Код программы

Откройте RIDE и создайте новый проект. Напишите C код и сохраните как act05.c файл (см. рис. 4.8)::

```
/*---------------------------------------*/
// Program     : Basic driving motor
// Description: Drive motors on forward and backward for checking direction
// Filename    : act0701.c
// C compiler : RIDE 51 V6.1
/*---------------------------------------*/
#include <C51ac2.h>          // Declare T89C51AC2's register
#include <sound_robo51.h>    // Declare Sound library
#include <dc_motor.h>        // Declare DC motor library
#include <delay_robo51.h>    // Declare Delay library

void main()
{
   beep();              // Beep for beginning
   motor_fd(1,100);     // Drive motor channel 1 forward
   motor_bk(2,100);     // Drive motor channel 2 backward
   delay_ms(10000);     // Delay 10 seconds for obseration motor turning
   motor_stop(all);     // Stop all motors
   beep();              // Beep for finishing
   while(1);            // Endless loop
}
```

Рисунок 4.8 – Код программы

Добавьте проект к HEX файлу.

Нажмите on на Robo-51 и загрузите на него HEX файл. Выключите питание и отключите кабель загрузки.

Поднимите робота над полом. Нажмите power-on и выберите режим RUN. Соблюдайте последовательность операций.

Robo-51 будет издавать звуковой сигнал, сразу же после этого коробка передач двигателя на выход M-1 поворачиваются вперед - светодиодный индикатор горит зеленым цветом. Мотор M-2 поворачивается назад - светодиод горит красным цветом. После этого оба двигателя запускаются в течение 10 секунд и останавливаются, звуковой сигнал снова заканчивается.

Программирование Robo-51

Библиотеки

Управления движением состоит из 4 основных файлов библиотеки: dc_motor.h, delay_robo51.h, и sound_robo51.h и lcd_robo51.h

dc_motor.h: библиотека приводного двигателя

motor_fd - запуск одного или двух двигателей в прямом направлении с 255 скоростным уровнем.

motor_bk - запуск одного или двух двигателей в обратном направлении с 255 скоростным уровнем.

motor_stop - торможение одного или двух двигателей (1, 2 и 3 парам. значение).

delay_robo51.h: библиотека времени задержки

delay_100us - время задержки в 100 мкс единицы (0 до 65535). Значение времени приблизительное.

delay_ms - время задержки в 1 миллисекунду блока (0 до 65535). Значение времени приблизительное.

sound_robo51.h: звуковая библиотека

sound - создает звуковой сигнал.

beep - создает звуковой сигнал на частоте 500 Гц длиной 0,1 секунда.

lcd_robo51.h: LCD интерфейс библиотеки

lcd_command - отправляет командный байт в ЖК-модуль.

lcd_text - отправляет текстовый байт в ЖК-модуль.

lcd_init – инициализирует LCD модуль для 4-битного режима интерфейса.

lcd_putstr - отправляет сообщение на ЖК-модуль.

inttolcd - преобразует числа в числовой текст и отправляет на модуль ЖК-дисплея.

Функции движения

Есть 5 движений робота: вперед, назад, влево, вправо и тормоз. Вы можете изменить и улучшить dc_motor.h библиотеки и добавить больше функций.

run_fd – перемещение вперед (см. рис. 4.9):

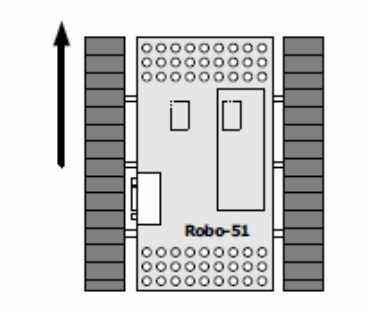


Рисунок 4.9 – Перемещение вперед

Эта функция будет вести оба двигателя в прямом направлении. Параметр этой функции time_spin. Это длительность движения вперед в единицу миллисекунды.

С код этой функции приведен ниже:

```
#define pow 200
void run_fd(int time_spin)
{
motor_fd(2,pow); // Motor channel 2 forward
motor_fd(1,pow); // Motor channel 1 forward
delay_ms(time_spin);// Delay time for robot driving forward
}
```

run_bk– перемещение назад (см. рис. 4.10):

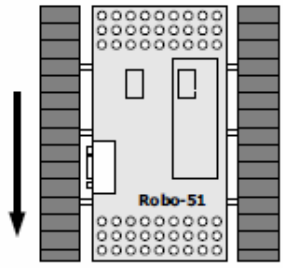


Рисунок 4.10 – Перемещение назад

Эта функция будет вести оба двигателя в обратном направлении. Параметр этой функции time_spin. Это длительность обратного движения в единицу миллисекунды.

С код этой функции приведен ниже:

```
#define pow 200
void run_bk(int time_spin)
{
motor_bk(2,pow); // Motor channel 2 backward
motor_bk(1,pow); // Motor channel 1 backward
```

```
delay_ms(time_spin);// Delay time for robot driving backward
}
```

turn_left: движение налево (см. рис. 4.11):

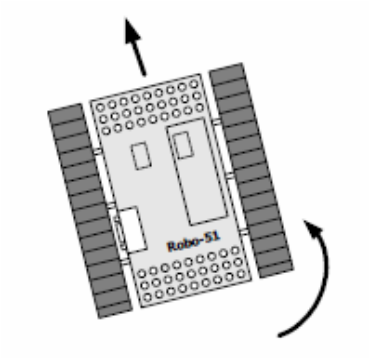


Рисунок 4.11 – Перемещение влево

Эта функция будет вести оба двигателя в разном направлении. М-2 двигатель - вперед и М-1 двигатель - в обратном направлении. Параметр этой функции time_spin. Это длительность движения влево в единицу миллисекунды.

С код этой функции приведен ниже:

```
#define pow 200
void turn_left(int time_spin)
{
motor_fd(2,pow); // Motor channel 2 forward
motor_bk(1,pow); // Motor channel 1 backward
delay_ms(time_spin);// Delay time for robot spining turn left
}
```

turn_ right: движение направо (см. рис. 4.12):

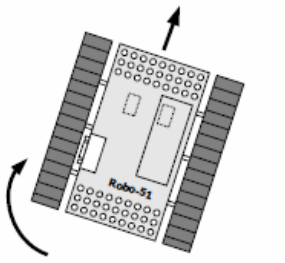


Рисунок 4.12 – Перемещение направо

Эта функция будет вести оба двигателя в разном направлении. М-2 двигатель – в обратном направлении и М-1 двигатель - в направлении прямо. Параметр этой функции time_spin. Это длительность движения вправо в единицу миллисекунды.

С код этой функции приведен ниже:

```
#define pow 200
void turn_right(int time_spin)
{
motor_bk(2,pow); // Motor channel 2 backward
motor_fd(1,pow); // Motor channel 1 forward
delay_ms(time_spin); // Delay time for robot spining turn right
}
```

stop: останковка движения

Эта функция, тормозящая действия робота.

С код этой функции приведен ниже:

```
void stop(void)
{
motor_stop(3); // Brake both motor outputs
// (call from dc_motor.h library file)
}
```

Простые движения robo51

Откройте RIDE и создайте новый проект. Напишите С код и сохраните как act0801.c файл (см. рис. 4.13). Добавьте проект к HEX файлу.

```
/*-----------------------------------------*/
// Program : Basic Move
// Description: Robo-51 movement by simple fuction
// Filename: act0801.c
// C compiler : RIDE 51 V6.1
/*-----------------------------------------*/
#include <C51ac2.h>    // Header include register of P89C51AC2
#include <lcd.h>       // Module function LCD display
#include <dc_motor.h>  // Module function drive DC motor
#include <sound.h>     // Module function drive sound
#include <delay.h>     // Module function delay time
#define pow 200        // Define constant for power drive DC motor

void run_fd(int time_spin)
{
   motor_fd(2,pow);    // Motor channel 2 forward
   motor_fd(1,pow);    // Motor channel 1 forward
   delay_ms(time_spin);// Delay time for robot driving forward
}

void run_bk(int time_spin)
{
   motor_bk(2,pow);    // Motor channel 2 backward
   motor_bk(1,pow);    // Motor channel 1 backward
   delay_ms(time_spin);// Delay time for robot driving backward
}

void main()
{
   while(1)            // Infinite loop
   {
     run_fd(2000);     // Run forward 2 seconds
     beep();           // Sound 1 time
     run_bk(2000);     // Run backward 2 seconds
     beep();           // Sound 1 time
   }
}
```

Рисунок 4.13 – Код программы

Нажмите on на Robo-51 и загрузите на него HEX файл. Выключите питание и отключите кабель загрузки.

Установите робота на пол. Нажмите power-on и выберите режим RUN. Соблюдайте последовательность операций.

Robo-51 движется вперед 2 секунды и назад 2 секунды. Каждое изменение движения сопровождается звуковым сигналом.

Если действие некорректное, то нужно поменять кабельное соединения двигателя для RBX-51AC2 платы. Изменять до исправления. При движении вперед оба индикатора двигателя горят (светятся) зеленым и красным цветом при обратном движении.

Создайте новый проект. Напишите C код и сохраните как act0802.c файл (см. рис. 4.14). Добавьте проект к HEX файлу.

Нажмите on на Robo-51 и загрузите на него HEX файл. Выключите питание и отключите кабель загрузки.

Установите робота на пол. Нажмите power-on и выберите режим RUN. Соблюдайте последовательность операций.

Robo-51 движется по заданной программе и показывает направление движения сообщением на ЖК-экран.

(1) Перемещается вперед 2 секунды и показывает сообщение Forward.

(2) Поворачивает налево 0,5 секунды и показывает сообщение Turn Left.

(3) Перемещается вперед 2 секунды и показывает сообщение Forward снова.

(4) Поворачивает налево 0,5 секунды и показывает сообщение Turn Right.

(5) Возвращается к шагу (1) - (4).

```
/*-----------------------------------------*/
// Program     : Basic Move -2
// Description: Robo-51 movement with display operation on LCD
// Filename    : act0802.c
// C compilar  : RIDE 51 V6.1
/*-----------------------------------------*/
#include <C51ac2.h>         // Header include register of P89C51AC2
#include <lcd_robo51.h>     // LCD Module function
#include <dc_motor.h>       // DC motor driving module function
#include <sound_robo51.h> // Sound ganaration module function
#include <delay_robo51.h> // Delay time module function
#define pow 200             // Define constant for power drive DC motor

void run_fd(int time_spin)
{
   motor_fd(2,pow);         // Motor channel 2 forward
   motor_fd(1,pow);         // Motor channel 1 forward
   delay_ms(time_spin);     // Delay time for forward movement
}
void run_bk(int time_spin)
{
   motor_bk(2,pow);         // Motor channel 2 backward
   motor_bk(1,pow);         // Motor channel 1 backward
   delay_ms(time_spin);     // Delay time for backward movement
}

void turn_left(int time_spin)
{
   motor_fd(2,pow);         // Motor channel 2 forward
   motor_bk(1,pow);         // Motor channel 1 backward
   delay_ms(time_spin);     // Delay time of turn left
}
void turn_right(int time_spin)
{
   motor_bk(2,pow);         // Motor channel 2 backward
   motor_fd(1,pow);         // Motor channel 1 forward
   delay_ms(time_spin);     // Delay time of turn right
}
void main()
{
   lcd_init();                          // Initial LCD module
   while(1)                             // Infinite loop
   {
     lcd_command(0x01);                 // Clear display
     lcd_putstr(1line1,"  Forward");    // Show message
     run_fd(2000);                      // Run forward 2 seconds

     lcd_command(0x01);                 // Clear display
     lcd_putstr(1line1,"  Turn Left"); // Show message
     turn_left(500);                    // Turn left 0.5 second

     lcd_command(0x01);                 // Clear display
     lcd_putstr(1line1,"  Forward");    // Show message
     run_fd(2000);                      // Run forward 2 seconds

     lcd_command(0x01);                 // Clear display
     lcd_putstr(1line1,"  Turn Right");// Show message
     turn_right(500);                   // Turn Right 0.5 second
   }
}
```

Рисунок 4.14 – Код программы

Задание по выполнению лабораторной работы:

1. Изучить основные принципы управления простыми движениями мехатронной системы IE-ROBO-51.
2. Осуществить программным управлением движение робота вперед.
3. Осуществить программным управлением движение робота назад.

4. Осуществить программным управлением движение робота налево.

5. Осуществить программным управлением движение робота направо.

Контрольные вопросы:

1. Понятия о мехатронных модулях движения.
2. Классификация приводов.
3. Способы управления приводами МС.

Лабораторная работа №5

Управление скоростью движения мехатронной системы IE-ROBO-51

Цель работы:

Изучить принципы управления скоростью движения мехатронной системы IE-ROBO-51, написать тестовую программу для управления скоростью движения МС IE-ROBO-51.

Основные теоретические сведения:

Используйте знания, полученные при выполнении лабораторной работы №4.

Задание по выполнению лабораторной работы:

1. Изучить способы управления скоростью движения МС.
2. Написать тестовую программу для управления скоростью движения лабораторным макетом.
3. Запустить тестовую программу для управления скоростью движения МС IE-ROBO-51.

Контрольные вопросы:

1. Способы управления скоростью движения МС.
2. Основные виды модуляции.
3. Управление скоростью движения с помощью широтно-импульсной модуляции.
4. Техническая реализация широтно-импульсной модуляции.

Лабораторная работа №6

Очувствление мехатронной системы IE-ROBO-51.
Чтение аналогового сигнала.
Бесконтактное обнаружение объектов.

Цель работы:

Изучить методы программного обеспечения для чтения аналогового сигнала, а также интерфейс аналогового датчика сигнала и программируемые методы считывания для бесконтактного обнаружения объектов.

Основные теоретические сведения:

Чтение коммутатора опросным методом

1 Откройте RIDE и создайте новый проект. Сделайте С код (рис. 6.1) и сохраните файл, как act07.c. Сборка проекта в HEX файле.

2 Включите Robo-51 и скачайте шестнадцатеричный файл к нему. Выключите питание и отсоедините загрузочный кабель.

3 Подключите ZX-01 переключатель модуля P3.2 на RBX-51AC2 плату контроллера от Robo-51.

4 Включите питание. Программа будет работать. Попробуйте нажать на переключатель и наблюдать операцию.

LCD модуль показывает сообщение Input P3.2 Read в строке выше и показывает состояние датчика («0» или «1») в нижней строке. Если коммутатор не нажат, статус показывает «1», а если коммутатор нажат – «0».

```
/*-------------------------------------------------------------------------*/
// Program     : Reading digital input
// Description: Reading digital signal from switch by polling method
// Filename    : act0901.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------*/
#include <C51ac2.h>          // Header include register of P89C51AC2
#include <lcd_robo51.h>      // Module function LCD display
#include <delay_robo51.h> // Module function Delay
sbit sw1 = P3^2;             // Define variable for P3.2
void main(void)
{
   sw1 = 1;                  // Write logic "1" for setting P3.2 to input
   lcd_init();               // LCD initialize
   lcd_putstr(line1," Input P3.2 Read"); // Show "Input P3.2 Read" on LCD
   while(1)                  // Loop
   {
     inttolcd(0xC7,sw1,Dec); // Read P3.2 data to convert and send to
                             // LCD for displaying on lower line
   }
}
```

Рисунок 6.1 – Код программы для чтения цифрового датчика активности из Robo-51

Описание программы:
Микроконтроллер проводит опрос, чтобы прочитать статус модуля P3.2. Он получает статус в численном значении для преобразования в ASCII код и отображения на LCD модуль.

Чтение коммутатора с перерывами

1 Откройте RIDE и создайте новый проект. Сделайте C код (рис. 6.2) и сохраните как файл act1001.c. Сборка проекта в HEX файле.

```
/*-----------------------------------------*/
// Program     : Reading digital input
// Description: Reading digital signal from switch input board by interrupt
// Filename    : act08.c
// C compiler : RIDE 51 V6.1
/*-----------------------------------------*/
#include <C51ac2.h>          // Header include register of P89C51AC2
#include <lcd_robo51.h>      // Module function LCD display
#include <delay_robo51.h>    // Module function Delay
unsigned int count=0;        // Declare count variable
void service_external0(void) interrupt 0 // External interrupt service routine
{
   lcd_clear();                          // Clear display
   lcd_putstr(line1," Count:");          // Show "Count" message
   count = 0;                            // Clear counter
}
void main(void)
{
   EX0 = 1;                  // Enable external interrupt on P3.2 (EX0)
   IT0 = 1;                  // Select active edge of input signal at P3.2
   EA = 1;                               // Enable global interrupt
   lcd_init();                           // LCD initialize
   lcd_putstr(line1," Count:");          // Show  " Count:"
   while(1)                              // Loop
   {
     inttolcd(0x88,count,Dec);           // Show counter on LCD
     delay_ms(100);                      // Delay 0.1second
     count++;                            // Increase counter
   }
}
```

Рисунок 6.2 – Код C для чтения сенсора методом перерывов

Описание программы:

Программа начинает подсчет от 0 до 65,535 каждые 0,1 секунды. Значение счетчика хранится в переменной количество (count) и отправляется для отображения на LCD модуль. Если коммутатор в положении контакта P3.2 нажат во время подсчета голосов. Прерывание произошло. В заголовке программы, включить внешнее прерывание на P3.2 или EX0. Если прерывание произошло, процессор (CPU) будет переходить на процедуру обслуживания прерывания в service_external0.

Процедура прерывания, начинается с чистым экраном командного модуля LCD Count: Показать на LCD экране и сбросить количество счета в значение 0 (Count = 0). После этого CPU будет возвращаться к основной программе, чтобы начать снова.

2 Запустить Robo-51и скачать шестнадцатеричный (HEX) файл к нему. Выключите питание и отключите загрузочный кабель.

3 Подключите коммутатор модуля ZX-01 в P3.2 на плату контроллера RBX-51AC2 Robo-51.

4 Включите питание. Программа будет работать. Попробуйте нажать на переключатель (коммутатор) и наблюдать операцию.

При запуске программы, LCD модуль показывает подсчет значений. Count: 0 → … → Count: 65535

Если кнопка нажата, значение обнуляется и подсчет начинается заново.

Перед нажатием кнопки (коммутатора) Count : 100 или другое значение.
После нажатия кнопки (коммутатора) Count: 0 и считать снова.

Аналоговый интерфейсов датчика

Robo-51 имеет 6 аналоговых входов для сопряжения с аналоговыми датчиками. Аналого-цифровой модуль преобразователя в микроконтроллер платы контроллера может принять напряжение более +3 В. Но с аттенюатором схема может помочь плате контроллера получить напряжение +5 В. Таким образом, Robo-51 может взаимодействовать со многими аналоговыми датчиками. Разрешающая способность 10. Это означает, что вы можете получить результат в диапазоне от 0 до 1023. Всего 1024 значения (рассчитывается по формуле от 210 = 1024).

Математическая связь между Vin (аналоговый вход напряжения) с преобразованием данных (A) от аналого-цифрового преобразователя может быть показана ниже:

$$Vin=\frac{5}{1023}*A, \qquad (6.1)$$

где 5 - напряжение питания +5 В;
1023 - максимальное значение преобразования данных.

Все 6 аналоговых входов Robo-51 будут подключены к аналоговому порту T89C51AC2 см. табл. 6.1.

Таблица 6.1 – Подключение аналоговых входов к порту

Аналоговый вход	Канал вывода порта
0	P1.0
1	P1.1
2	P1.2
3	P1.3
4	P1.4
5	P1.5

Управление регистра от конвертера A / D

1. ADCF: ADC Регистр конфигурации в табл. 6.2.

Таблица 6.2 – Работа регистра конфигурации

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

Это 8-битный регистр. Адрес является F6H. Этот регистр устанавливает использование P1.x как вход A / D конвертера и использует P1.x как стандартный цифровой порт ввода / вывода.

2. ADCON: ADC регистр управления в табл. 6.3.

Таблица 6.3 – Работа регистра управления

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
-	PSIDLE	ADEN	ADEOC	ADSST	SCH2	SCH1	SCH0

Это 8-битный регистр. Адрес является F6H. Подробностями каждого бита являются:

PSIDLE (бит 6): Псевдорежим ожидания (Best Precision).

"0" – Очистить, чтобы конвертировать без режима ожидания.

"1" - Установить и поставить в режим ожидания во время преобразования.

ADEN (бит 5): включение / режим ожидания.

"0" - режим ожидания (рассеиваемая мощность 1 мВт).

"1" - включить ADC.

ADEOC (бит 4): Конец преобразования.

"0" - модуль АЦП еще в пересчете.

"1" - ADC преобразование готово.

Устанавливается аппаратно, когда ADC результат готов для чтения и может генерировать прерывание, должен быть сброшен программно.

ADSST (бит 3): Начало и статус

"0" - сбрасывается аппаратно после завершения преобразования.

"1" - Начало A / D преобразования.

SCH2: 0 (бит 2-0): выбор канала для преобразования в табл. 6.4.

Таблица 6.4 – Описание работы SCH2

SCH2	SCH1	SCH0	Analog input channel
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN6
1	1	1	AN7

3. IEN1: регистр разрешения прерывания

Это 8-битный регистр. Адрес является E8h. Существует только один бит в использовании A / D конвертера. Это в ЕАДС (ADC прерываний бит – бит 1). Прерывание произошло, когда модуль АЦП конвертировал сигнал полного и бит ADEOC установлен. Если этот бит не установлен, то прерывания быть не может.

Вектор прерывания ADC адресу 0043H. Прерывание номер 8, когда происходит прерывание с программы Си.

4. ADDH и ADDL: ADC регистры данных

Регистры, которые хранят данные в результате преобразования. ADDH адрес F5H и F4H для ADDL регистра.

ADDH регистр хранит 8 верхних битов (от бит 9 до 2) преобразования результата.

ADDL регистр хранит 2немного нижних битов (биты 1 и 0), результат преобразования.

Чтение A/ D преобразования

1. Преобразование проверки

Это чтение можно резюмировать как:

(1)Установить порт АЦП через ADCF регистр. Пример:

(1)ADCF = 0x01; // Select AN0
(2)ADCF = 0x03; // Select AN0 and AN1
(3)ADCF = 0x84; // Select AN7 and AN2
(4)ADCF = 0xFF; // Select all analog input

(2)Установить ADEN для АЦП. ADEN является 5 битом для ADCON регистра.

Пример кода C:

ADCON = 0x20;

(3)Выберите вход через чтение SCH2 в SCH0 бит регистра ADCON.

Пример кода C:

```
ADCON &= 0xF8; // Clear all SCH2 to SCH0 bits
ADCON |= channel; // Select input with the value of
        // channel. The value is 0 to 7.
```

Если выбрать читать вход AN3, код C будет:

```
ADCON &= 0xF8;
      ADCON |= 3;
```

(4)Установите ADSST, чтобы начать преобразования. ADSST является 3 битом регистра ADCON.

После завершения преобразования, ADSST очистится автоматически. Пример кода C :

ADCON |= 0x08;

(5)После шага (4), модулю АЦП нужно время, чтобы преобразовать аналоговый сигнал в цифровые данные и хранить данные в ADDH и ADDL регистрах. После завершения преобразования, ADEOC (бит 4 ADCON регистре) устанавливается автоматически. Таким образом, программирование C должно сделать программу на выбор статуса этого бита. Когда бит ADEOC установлен, то это означает, что преобразование закончилось. Нужно очистить ADEOC бит перед чтением преобразования данных. Образец кода:

```
while((ADCON & 0x10) != 0x10);
// Wait until ADEOC flag is set.
ADCON &= 0xEF; // Clear ADEOC flag before
        // read the result.
```

(6)Чтение 10-разрядного преобразования данных из ADDH и ADDL регистров

Данные в регистре ADDH 8-битный верхних или бит 9 к 2.

Данные в регистре ADDL 2 нижних бита или 1 и 0.

При чтении полных 10-битов, переменная должна заявить целое..Пример кода:

```
int data; // Declare the result variable as integer.
data = (ADDH<<2) + ADDL;
        // Reda data to store in dat variable
```

2. Чтение с прерыванием

Это чтение можно резюмировать как:

(1) Установить порт входа АЦП (ADC) через ADCF регистр.

(2) Установить ADEN для АЦП.

(3) Подключение глобальных прерываний и прерывания АЦП с множеством EA и EADC.

Образец кода:

```
EA =1; // Enable global interrupt
EADC = 1; // Enable ADC interrupt.
```

(4) Выберите вход через чтение SCH2 в бит SCH0 в регистре ADCON.

(5) Установите ADSST, чтобы начать преобразования.

(6) После шага (4), модулю АЦП нужно время, чтобы преобразовать аналоговый сигнал в цифровые данные и хранить данные в ADDH и ADDL регистре. После завершения преобразования, прерывание произошло. Процессор будет перескакивать на выполнение прерывания. В этой стандартной программе должен очиститься первый бит ADEOC. После этого прочитается 10-битный результат преобразования. Образец процедуры прерывания показан ниже:

```
int adc_dat; // Declare the conversion variable
void ADC_service() interrupt 8 // Interrupt number is 8.
{
EADC = 0; // Disable interrupt
ADCON &= 0xEF; // Clear ADEOC bit.
adc_dat = (ADDH<<2)+(ADDL);
// Read data to store at adc_dat
ADCON |= 0x48; // Set ADEN and ADSST bit
// for next conversion
EADC = 1; // Enable interrupt
// for next conversion.
}
```

3. Чтение функции A/ D конвертера

Из всего шагов, которые описываются в предыдущем разделе, можно сделать подходящую функцию для чтения A / D преобразования и сохранения в файл adc_robo51.h библиотеки C: \ RIDE \ INC. Полный исходный код показан на рис. 6.3. Пример:

```
int dat; // Declare integer variable for getting data from
// analog function
data = analog(2); // Read the analog input 2 (P1.2) and store to dat
// variable.
```

```
/*-------------------------------------------*/
// Program : Read Analog input
// Description: Read Analog input function
// Filename: adc_robo51.h
// C compiler : RIDE 51 V6.1
/*-------------------------------------------*/
int analog(char channel)
{
   int adc_dat = 0;     // For keep input analog data
   ADCF = 0x3F;         // Setup config for adc bit input convert(port1) (P1.0-
P1.5)
   ADCON = 0x20;                // Enable ADC
   ADCON &= 0xF8;               // clear SCH bit (SCH 0:1:2 = 0)
   ADCON |= channel;            // select convert/read by channel config.

   ADCON |= 0x08;               // start convert (set bit ADSST)
   while((ADCON & 0x10)!= 0x10);// wait for EOF to set
   ADCON &= 0xEF;               // clear bit ADEOC
   adc_dat = (ADDH<<2)+(ADDL);  // read adc data and convert
   return(adc_dat);             // Return result of ADC 10 bit
}
```

Рисунок 6.3 – Исходный код для функции чтения A / D преобразования

Чтение аналогового сигнала

1. Откройте RIDE и создайте новый проект. Сделайте C код (рис. 6.4) и сохраните как файл act1101.c. Сборка проекта в HEX файле.

2. Запустите Robo-51и HEX файл. Выключите питание и отключите загрузочный кабель.

3. Подключите модуль ZX-потенциометра к входу AN0 на RBX-51AC2 контроллере платы Robo-51.

4. Включите питание. Программа будет работать. Попробуйте включить вал потенциометра и наблюдать операции.

Результат преобразования показан на LCD модуле следующей регулировки потенциометра. Данные от 0 до 1023. Однако максимальный результат может не доходить до 1023, т. к это зависит от уровня заряда батареи.

```
/*-------------------------------------------------------------*/
// Program : Reading digital data from A/D convertion
// Filename: act1101.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------*/
#include <C51ac2.h>          // Header include register of P89C51AC2
#include <lcd_robo51.h>      // Module function LCD display
#include <delay_robo51.h>    // Module function Delay
#include <adc_robo51.h>      // Module function analog to digital converter
void main(void)
{
   unsigned int val;         // Declare analog data variable,
                             // integer type recommended
   lcd_init();               // Initial LCD module
   while(1)                  // Looping
   {
     lcd_putstr(line1," Analog CH0 Read");  // Show title message
     val = analog(0);        // Read digital data from conversion to variable
     inttolcd(0xC7,val,Dec); // Convert digital data to show on LCD display
     delay_ms(1000);         // Delay 1 second
     lcd_clear();
   }
}
```

Рисунок 6.4 – Код программы для чтения A / D преобразования

Задание по выполнению лабораторной работы:

1) Чтение и включение сенсора с помощью метода использования цифрового интерфейса.
2) Чтение и включение сенсора методом прерывания.
3) Чтение аналогового сигнала.

Контрольные вопросы:

1) Очувствление МС.
2) Основные уровни интеграции МС.
3) Эволюция МС.

Лабораторная работа №7

Движение робота по заданной траектории
Чтение датчика отслеживания линий.
Движение вдоль чёрной линии.
Программное задание траектории движения

Цель работы:

Управление роботом по заданной траектории: изучение метода чтения датчика отслеживания линий, движения вдоль линии и программное задание траектории.

Основные теоретические сведения:

Тестирование GP2D120

1. Откройте RIDE и создайте новый проект. Сделайте С код (рис. 7.1) и сохраните как файл act1201.c. Сборка проекта в HEX файле.
2. Запустите Robo-51и HEX файл. Выключите питание и отключите загрузочный кабель.
3. Включите питание. Запустите программу, выбрав переключатель режимов в положение RUN. Поместите некоторые объекты на передней панели модуля GP2D120. Соблюдайте операции LCD.
4. Отрегулируйте расстояние объекта от датчика GP2D120 и наблюдайте за результатом.

Из тестирования, вы увидите, что GP2D120 может обнаружить объект в диапазоне от 4 до 32см.

```
/*-------------------------------------------*/
// Program : GP2D120 demo
// Description: Get distance from GP2D120
// Filename: act1201.c
/*-------------------------------------------*/
#include <C51ac2.h>        // Header include register of P89C51AC2
#include <lcd_robo51.h>       // Module function LCD display
#include <delay.h>         // Module function Delay
#include <gp2d120.h>       // Module function get distance from GP2D120
void main(void)
{
 unsigned int val;         // Declare distance data variable(cm unit),
                           // integer type recommended
 lcd_init();               // Initial LCD module
 while(1)                  // Looping
 {
  val = dist(1);           // Read digital data from conversion to variable val
  if(val>=4 && val<=32)
  {
    lcd_putstr(line1,"Distance:     cm  ");
                           // Show message "Distance      cm" on LCD module
    inttolcd(0x8A,val,Dec); // Convert digital data to show on LCD display
  }
  else
  {
   lcd_putstr(line1,"Out of Range      ");  // Show message on LCD module
  }
  delay_ms(1000);          // Delay 0.1 second for display value
  lcd_clear();             // Clear display
 }
}
```

Рисунок 7.1 – Демонстрация программы GP2D120

Бесконтактный робот обнаружения объекта

1. Откройте RIDE и создайте новый проект. Сделайте C код (см. приложение A) и сохраните как файл act1301.c. Сборка проекта в HEX файле.

2. Запустите Robo-51и HEX файл. Выключите питание и отключите загрузочный кабель.

3 Поставьте робота на пол.

4 Включите питание. Запустите программу, выбрав переключатель режимов в положение RUN.

5. Попробуйте поставить любой объект в передней части робота и посмотрите на его работу.

Робот будет проверять расстояние до объекта спереди.
Если нет никаких препятствий, робот будет продолжать двигаться вперед.
Если найден объект, он будет двигаться назад, поворот влево и будет двигаться вперед.

Чтение значения IR отражателя

1. Откройте RIDE и создайте новый проект. Сделайте C код (рис. 7.2) и сохраните как файл act1401.c. Сборка проекта в HEX файле.

2. Запустите Robo-51и HEX файл. Выключите питание и отключите загрузочный кабель.

3. Подключите левый ZX-03 инфракрасный модуль отражателя для AN5 и прямо на входе AN4, на RBX-51AC2 платы контроллера от Robo-51.

4. Поставьте робота на черную область. Включите питание. Программа будет работать. Наблюдайте работу на LCD модуле. После этого измените область белым цветом и запустить снова.

На LCD модуле показывается значение входа AN4. Если значение по ссылке; 400, Robo-51 будет управлять звуковым сигналом. Держите эту операцию для использования в соответствии с линией деятельности.

```
/*-------------------------------------------------------------------------------*/
// Program     : Reading IR reflect sensor
// Description: Reading white and black surface from IR reflect sensor
// Filename    : act1401.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------------*/
#include <C51ac2.h>          // Header include register of P89C51AC2
#include <lcd_robo51.h>      // Module function LCD display
#include <delay.h>           // Module function Delay
#include <adc_robo51.h>      // Module function Analog to digital converter
#include <sound_robo51.h>    // Module function Sound
#define ref 400              // Define reference value as 400
void main(void)
{
   unsigned int reflect_dat;        // Declare reflect_dat for keeping analog
                                    // data from IR reflector
   lcd_init();                      // Initailize LCD module
   lcd_putstr(line1,"Reflect  CH4"); // Show message "Reflect CH4" on LCD module
   while(1)                         // Looping
   {
      reflect_dat = analog(4);      // Read analog value from IR reflect sensor
                                    //and collect data at reflect_dat
      inttolcd(0xC7,reflect_dat,Dec); // Display analog value on LCD module
      if(reflect_dat>ref)           // Compare analog value with reference
      beep();                       // Beep if reflect_dat > reference
      delay_ms(500);                // Delay 0.5 second
   }
}
```

Рисунок 7.2 – Программа для чтения отражателя инфракрасного датчика

Robo-51 находит черную линию

Представляется нахождение и обнаружения черной линии Robo-51. Робот
всегда двигаться вперед, пока не обнаружить черную линию. После обнаружения, робот остановится и будет управлять звуковым сигналом (рис. 7.3.).

1. На рис. 7.3, показана черная линия шириной 1 дюйм и длиной 8 дюймов на белой поверхности.

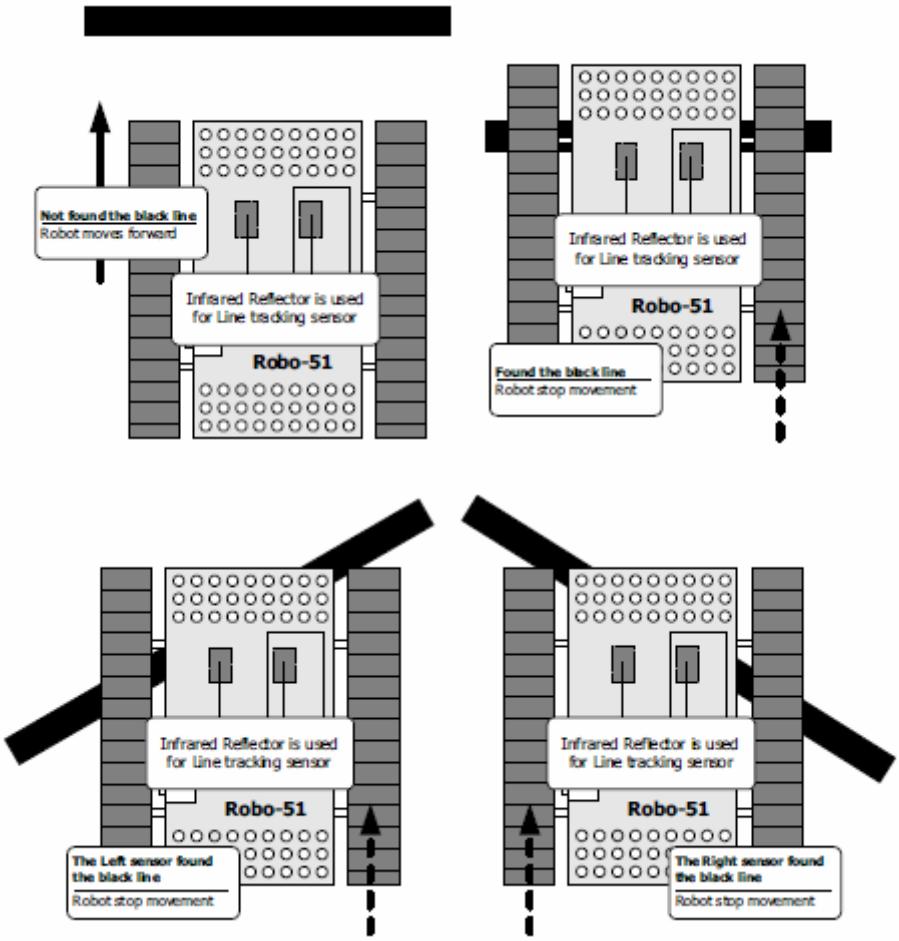


Рисунок 7.3 – Демонстрация операций нахождения черной линии Robo-51

2. Откройте RIDE и создайте новый проект. Сделайте C код (см. приложение Б) и сохраните как файл act1501.c. Сборка проекта в HEX файле.

3. Запустите Robo-51и HEX файл. Выключите питание и отключите загрузочный кабель.

4. Место робота над белой поверхностью определяется следующим образом (место одной позиции тестируется первой и положение изменяется к тестированию в следующем):

(A) Правый угол с черной линией, начиная с шага A11.1;

(B) 45 градусов слева с черной линией;

(C) 45 градусов на правый с черной линией.

Включите питание на ходу. Обратите внимание на движение робота и LCD отображения.

Robo-51 управляет звуковым сигналом и движется вперед, чтобы найти черные линии. LCD модуль показывает результат обнаружения.

Любой датчик может обнаружить черные линии, LCD дисплей будет показывать слово "Линия", когда датчик сможет обнаружить линию.
После этого, происходит остановка движения Robo-51.

Robo-51 пин-понг

Представляется управление Robo-51 для перемещения в Zig-Zag формате. Черная линия является поворотным пунктом на рис. 7.4 ниже. Robo-51 движется вперед, пока не найдена черная линия, чтобы изменить направление. Таким образом, робот будет двигаться в пространстве между двумя черными линиями.

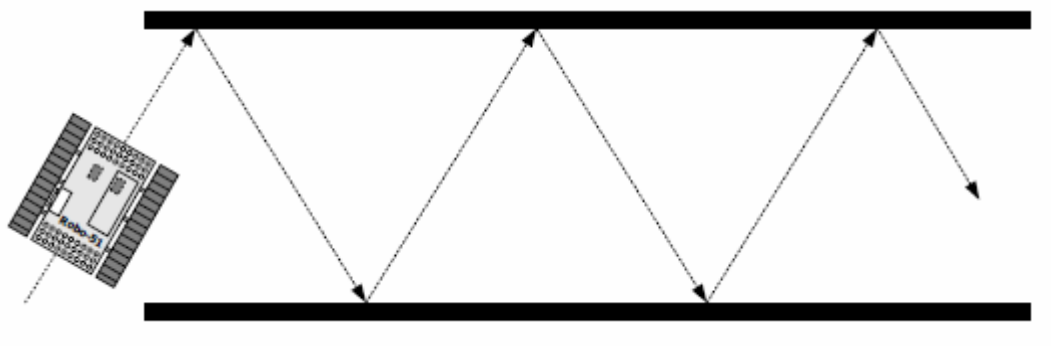


Рисунок 7.4 – Перемещение в формате Zig-Zag

Этапы программирования в этой деятельности:
(1) Найдите решение значения между белой поверхностью и черной линией.
(2) Прочитайте значения датчиков и сохраните для сравнения
(3) Проверьте состояние следующим образом:

Случай № 1: Оба датчика обнаруживают белую поверхность
Действие: Робот движется вперед.

Случай № 2: левый датчик обнаружил черную линию, а правый датчик обнаружил белую поверхность.
Действие: Робот движется с поворотом вправо на соответствующее значение для изменения направления, чтобы перейти к черной линии противоположной стороны.

Случай № 3: левый датчик обнаружил белую поверхность, но правый датчик обнаружил черную линию.
Действие: Робот движется с поворотом влево на соответствующее значение для
изменения направления, чтобы перейти к черной линии противоположной стороны.

Случай № 4: из 3 указанных выше условий.
Действие: управление роботом для перемещения следующего программного
результата.
(4) Вернуться к шагу (2).

Создание демонстрационных полей:

Подготовьте ровную белую поверхность размером 2 метра и более и 60 сантиметров шириной или больше. Придерживайтесь 1 дюйма ширины и 2 метра длинны параллельных черных линий. Расстояние s между линиями не менее 40 сантиметров.

Деятельность процедуры:

1. Откройте RIDE и создайте новый проект. Сделайте C код (см. Приложение B) и сохраните как файл act1601.c. Сборка проекта в HEX файле.
2. Запустите Robo-51 и HEX файл. Выключите питание и отключите загрузочный кабель.
3. Поместите Robo-51 в начальную точку. После демонстрация фигуры поверните робота до 45 градусов.
4. Включите питание и выберите режим выполнения.

В рамках этой программы, определяется время поворота в приближении до 1 секунды. Это вызывает угол поворота φ (см. рис. 7.5). Программист может отрегулировать значение времени, чтобы сделать подходящий угол для контроля траектории движения правильно.

Если значение времени больше, угол φ узкий. Это вызывает робота возвращается к тому же пути. С другой стороны, если значение времени меньше, робот должен выйти из линии или двигаться параллельными линиями не обнаруживая линии.

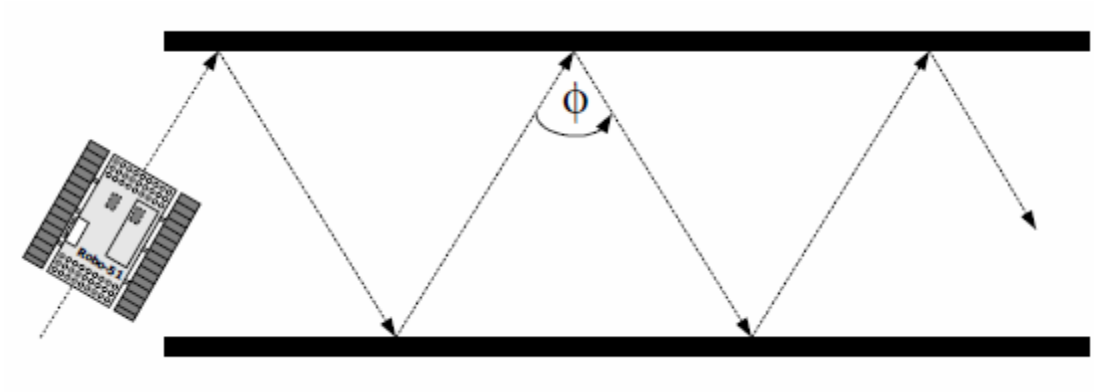


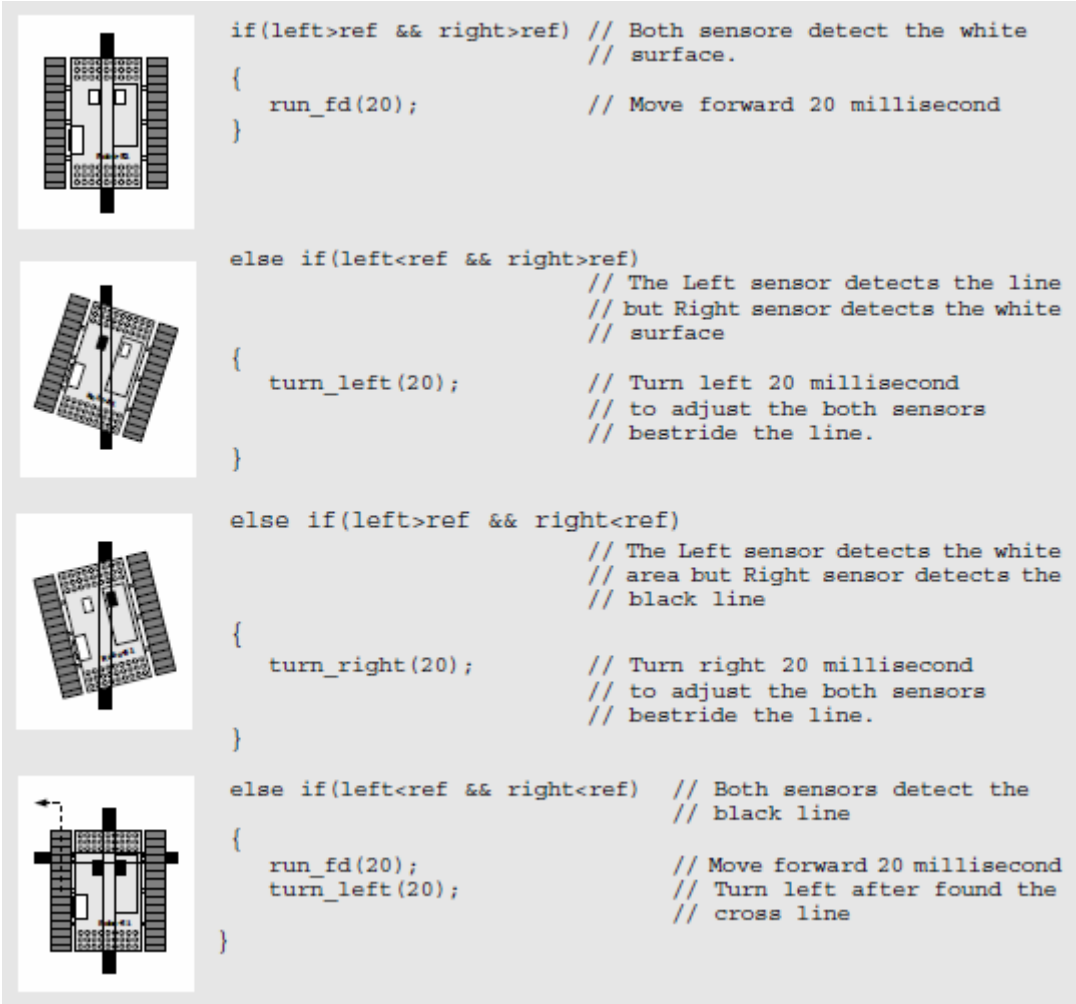
Рисунок 7.5 – Демонстрация угла поворота

Robo-51 отслеживает черную линию

1. Откройте RIDE и создайте новый проект. Сделайте C код (см. Приложение Г) и сохраните как файл act1701.c. Сборка проекта в HEX файле.
2. Запустите Robo-51 и HEX файл. Выключите питание и отключите загрузочный кабель.
3. Подготовьте демонстрационное поле сделать 1-дюймовый ширины черные линии на белой поверхности. Вы можете создать линию картины с вашей идеей.

4. Поместите Robo-51 над черной линией. Включите питание и выберите режим выполнения.
Соблюдайте операции.

Есть 4 условия для принятия решения в этой программе следующим образом, показанном на рис. 7.6.



```
if(left>ref && right>ref) // Both sensore detect the white
                          // surface.
{
   run_fd(20);            // Move forward 20 millisecond
}

else if(left<ref && right>ref)
                          // The Left sensor detects the line
                          // but Right sensor detects the white
                          // surface
{
   turn_left(20);         // Turn left 20 millisecond
                          // to adjust the both sensors
                          // bestride the line.
}

else if(left>ref && right<ref)
                          // The Left sensor detects the white
                          // area but Right sensor detects the
                          // black line
{
   turn_right(20);        // Turn right 20 millisecond
                          // to adjust the both sensors
                          // bestride the line.
}

else if(left<ref && right<ref)   // Both sensors detect the
                                 // black line
{
   run_fd(20);                   // Move forward 20 millisecond
   turn_left(20);                // Turn left after found the
                                 // cross line
}
```

Рисунок 7.6 – Программа отслеживания линий для Robo-51

Приложение А

```
/*-------------------------------------------------------------------------
--------------------*/
// Program : Robo-51 escape object by GP2D120 sensor
// Description: Robo-51 move by dc motor and detect object by GP2D120 sensor
// Filename : act1301.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------
--------------------*/
#include <C51ac2.h> // Header include register of P89C51AC2
#include <lcd_robo51.h> // Module function LCD display
#include <dc_motor.h> // Module function drive DC motor
#include <gp2d120.h> // Module function drive DC motor
#include <delay.h> // Module function delay time
#define pow 200 // Define constant for power drive DC motor
/***************** Function drive robot forward ******************************/
void run_fd(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot backward *****************************/
void run_bk(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn left ****************************/
void turn_left(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn right ***************************/
void turn_right(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/****************************** Main function ********************************/
void main()
{
unsigned int val; // Variable for keep distance
while(1) // Looping
{
val = dist(1); // Get distance from GP2D120
if(val<15) // Distance less than 15 cm?
{
run_bk(1000); // Backward 1 sec
turn_right(500); // Turn right 0.5 sec
}
else
{
run_fd(10); // Forward 10 ms
}
}
```

Приложение Б

```
/*-------------------------------------------------------------------------*/
// Program : Detect black line
// Description: Drive robot for finding the black line by IR reflect sensor
// Filename : act1501.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------*/
#include <C51ac2.h> // Header include register of T89C51AC2
#include <lcd_robo51.h> // LCD module function
#include <dc_motor.h> // DC motor module function
#include <sound_robo51.h> // Sound module function
#include <delay_robo51.h> // Delay time function
#include <adc_robo51.h> // ADC module function
#define pow 100 // Define constant for driving robot
#define ref 400 // Define constant the reference value for detection line
/***************** Function drive robot forward ******************************/
void run_fd(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot backward *****************************/
void run_bk(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn left ****************************/
void turn_left(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn right ***************************/
void turn_right(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/****************************** Main function *******************************/
void main()
{
int left=0,right=0; // Define variable for keep input analog value
lcd_init(); // Initial LCD module
beep(); // Sound beep 1 time

while(1) // Infinite loop
{
left = analog(5); // Read input analog channel 5 (connect left sensor)
right = analog(4); // Read input analog channel 4 (connect right sensor)
if(left<ref || right<ref) // If any sensor detect black line
{
motor_stop(all); // Stop robot
beep(); // Beep after any sensor detect line
```

```
lcd_clear(); // Clear LCD display
if(left<ref)
{
lcd_putstr(line1,”Left:>> Line”); // Left sensor detects line
}
else
{
lcd_putstr(line1,”Left:>> Floor”); // Left sensor detects floor
}
if(right<ref)
{
lcd_putstr(line2,”Right:>> Line”); // Right sensor detects line
}
else
{
lcd_putstr(line2,”Right:>> Floor”); // Right sensor detects floor
}
while(1); // Break program
}
else
{
run_fd(10); // Drive robot forward
}
}
}
```

Program description

When program start, Robo-51 drive a beep for beginning. Robo-51 polls to read the value from both analog inputs which connect the Infrared Reflectors. The value from Left sensor stores to left variable and Right sensor to right variable. Compare both value with the reference value; ref. This is comparative condition :

_ **left<ref**

“True” means the Left sensor detects the black line.

“False” means the Left sensor detects the white surface.

_ **right<ref**

“True” means the Right sensor detects the black line.

“False” means the Right sensor detects the white surface.

From condition testing of **if(left<ref || right<ref)**, if the result is **true** the meaning will be

Case #1 : **left<ref** is “true” and **right<ref** is “false”

Case #2 : **left<ref** is “false” and **right<ref** is “true”

Case #3 : **left<ref** is “true” and **right<ref** is “true”

Robo-51 move forward until detects the black line. It stop and drive a beep and reports the detected condition from Case #1 to #3 on LCD module. At the upper line, it shows the operation of Left sensor and bottom line shows the operation of Right sensor.

Пириложение В

```
/*-------------------------------------------------------------------------
-------------------*/
// Program : Ping-Pong Robot
// Description: Drive robot by black line reflection
// Filename : act1601.c
/*-------------------------------------------------------------------------
-------------------*/
#include <C51ac2.h> // Header include register of T89C51AC2
#include <dc_motor.h> // DC motor module function
#include <sound_robo51.h> // Sound module function
#include <delay_robo51.h> // Delay time function
#include <adc_robo51.h> // ADC module function
#define pow 250 // Define constant power drive robot
#define ref 300 // Define constant reference compare track line
int left=0,right=0; // Define variable for keep input analog value
/***************** Function drive robot forward ******************************/
void run_fd(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot backward ******************************/
void run_bk(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn left ******************************/
void turn_left(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn right ******************************/
void turn_right(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/****************************** Main function ********************************/
void main()
{
beep(); // Sound beep 1 time
while(1) // Infinite loop
{
left = analog(5); // Read input analog channel 5(left sensor)
right = analog(4); // Read input analog channel 4(right sensor)
if(left>ref && right>ref) // if both sensors detect white
{
run_fd(20); // Drive forward
else if(left<ref && right>ref) // if left sensor detects black,
// right sensor detects white
{
turn_right(1000); // Drive turn right
```

```
}
else if(left>ref && right<ref) // if left sensor detect white
// and right sensor detect black
{
turn_left(1000); // Drive turn left
}
else if(left<ref && right<ref) // if both sensors detect black
{
run_bk(20); // Drive backward
}
}
}
```

Приложение Г

```
/*-------------------------------------------------------------------------------------------------------------*/
// Program : Tracking line robot V1
// Description: Drive robot to tracking the black line
// Filename : act1701.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------------------------------------------*/
#include <C51ac2.h> // Header include register of T89C51AC2
#include <dc_motor.h> // DC motor module function
#include <sound_robo51.h> // Sound module function
#include <delay_robo51.h> // Delay time function
#include <adc_robo51.h> // ADC module function
#define pow 250 // Define constant power drive robot
#define ref 400 // Define constant reference compare track line
int left=0,right=0; // Define variable for keep input analog value
/***************** Function drive robot forward ******************************/
void run_fd(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot backward *****************************/
void run_bk(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn left ****************************/
void turn_left(int time_spin)
{
motor_fd(2,pow); // Drive left motor forward
motor_bk(1,pow); // Drive right motor backward
delay_ms(time_spin); // Delay for robot spin
}
/***************** Function drive robot turn right ***************************/
void turn_right(int time_spin)
{
motor_bk(2,pow); // Drive left motor backward
motor_fd(1,pow); // Drive right motor forward
delay_ms(time_spin); // Delay for robot spin
}
/****************************** Main function ********************************/
void main()
{
beep(); // Sound beep at once
while(1) // Infinite loop
{
left = analog(5); // Read input analog channel 5(connect left sensor)
right = analog(4); // Read input analog channel 4(connect right sensor)
if(left>ref && right>ref) // Both sensors detect white
{
run_fd(20); // Move forward
}
else if(left<ref && right>ref) // Left sensor detects black and
```

```
// Right sensor detects white
{
turn_left(20); // Turn left
}
else if(left>ref && right<ref) // Left sensor detects white and
// Right sensor detects black
{
turn_right(20); // Turn right
}
else if(left<ref && right<ref) // Both sensors detect black
{
run_fd(20); // Move forward
turn_left(20); // Turn left
}
}
}
```

Задание по выполнению лабораторной работы:

1) Изучение принципов управления движением робота по заданной траектории.

2) Составление тестовой программы и её запуска на лабораторном макете.

3) Управление движением робота вдоль черной линии, программное управление роботом «по коридору».

4) Программное задание траектории движения робота.

Контрольные вопросы:

1) Промышленные и мобильные роботы.

2) Мехатронные технологические машины.

3) Применение МС в автоматизации технического оборудования.

4) МС в автомобильном, морском, авиационном и др. видах транспорта.

Лабораторная работа №8

Робот с дистанционным управлением
Чтение команд дистанционного управления
Управление движением IE-ROBO-51 на ИК лучах

Цель работы:

Изучение работы робота с дистанционным управлением, дистанционное управление и чтение команд, управление движением IE-ROBO-51 на инфракрасных лучах.

Основные теоретические сведения:

Чтение ER-4 удаленным управлением данных

1. Откройте RIDE и создайте новый проект. Сделайте C код (см. приложение A) и сохраните как файл act1801.c. Сборка проекта в HEX файле.
2. Запустите Robo-51и HEX файл загрузки.
3. Положите 2 AA батареи в ER-4 пульт дистанционного управления.
4. Выберите переключатель режима на RBX-51AC2 платы контроллера от Robo-51 в рабочий режим.
5. Включите ER-4 направления инфракрасного модуля приемника на Robo-511. Нажмите любую кнопку на ER-4 пульте дистанционного управления. Соблюдайте операции.

Нажмите кнопку A сразу:
Светодиоды P3.5 на RBX-51AC2 включены.
Нажмите кнопку A удерживая ее:
Светодиодные P3.5 являются ВКЛ и ВЫКЛ.
Нажмите кнопку B сразу:
Светодиод P3.6 на RBX-51AC2 включен.
Нажмите кнопку B удерживая ее:
Светодиод P3.6 является включенным и выключение продолжится до отпускания кнопки.
Нажмите кнопку C сразу:
Светодиод P3.7 на RBX-51AC2 включен.
Нажмите кнопку C удерживая ее:
Светодиод P3.7 является включенным и выключение продолжается до отпускания кнопки.
Нажмите кнопку D сразу:
Управление звуковым сигналом.
Нажмите кнопку D удерживая ее:
Управление звуковым сигналом продлится до отпускания кнопки.

Описание программы:
Эта программа управления микроконтроллером, для получения данных Из меню ER-4 и контроля 3-светодиодов на P3.5to P3.7 порта и пьезо-динамик.
Remote.h файл библиотеки включены взаимодействие ER-4 Пульт дистанционного управления с get_remote и clear_remote функции.

Программа начинается с инициализации системы для приема последовательных данных с remote_init функции. После этого программа будет читать на кнопку данные из возвращаемого значения get_remote функции с переключателем регистра команд. В каждом случае имеет перерыв команды
очистить все буфера с clear_remote функции. После очистки, программа может работать продолжают получать новые кнопки.

Контроль ROBO-51

1. Откройте RIDE и создайте новый проект. Сделайте C код (см. приложение Б) и сохраните как файл act15.c. Сборка проекта в HEX файле.
2. Запустите Robo-51и HEX файл загрузки.
3. Выберите переключатель режима на RBX-51AC2 платы контроллера от Robo-51 в режиме RUN.
4. Включите питание. Используйте ER-4 для контроля Robo-51. Соблюдайте операции.

Работу Ribi-51 можно резюмировать следующим образом:

- Нажмите кнопку D

Robo-51 продолжает двигаться вперед до нажатия любой кнопки.

- Нажмите кнопку A

Robo-51 продолжает двигаться в обратном направлении до нажатия любой кнопки.

- Нажмите кнопку C

Robo-51 повернется налево. Если еще нажать на кнопку, Robo-51 будет продолжать поворачиваться налево до тех пор, пока кнопку не отпустят.

- Нажмите кнопку B

Robo-51 повернется направо. Если еще нажать на кнопку, Robo-51 будет продолжать поворачиваться направо до тех пор, пока кнопку не отпустят.

Приложение А

```
/*------------------------------------------------------------------------------------------------------------
------------------------------------------------------------*/
// Program : Test Infrared receiving
// Description: Control LED at P3.5,P3.6,P3.7 and sound by ER-4 Remote control
// Filename : act1801.c
// C compiler : RIDE 51 V6.1
/*------------------------------------------------------------------------------------------------------------
------------------------------------------------------------*/
#include <C51ac2.h> // Header include register of P89C51AC2
#include <delay_robo51.h> // Delay time module function
#include <remote.h> // ER-4 Remote Control function
#include <sound_robo51.h> // Sound module function
sbit led1 = P3^5; // Define LED on P3.5 port
sbit led2 = P3^6; // Define LED on P3.6 port
sbit led3 = P3^7; // Define LED on P3.7 port
void main()
{
remote_init(); // Initialize ER-4 remote control
while(1) // Looping
{
switch(get_remote()) // Check command from ER-4 remote control
{
case ‘a’ : led1 = 0; // LED P3.5 on
clear_remote(); // Clear command
break; // Out of from case
case ‘A’ : led1 = 1; // LED P3.5 off
clear_remote(); // Clear command
break; // Out of from case
case ‘b’ : led2 = 0; // LED P3.6 on
clear_remote(); // Clear command
break; // Out of from case
case ‘B’ : led2 = 1; // LED P3.6 off
clear_remote(); // Clear command
break; // Out of from case
case ‘c’ : led3 = 0; // LED P3.7 on
clear_remote(); // Clear command
break; // Out of from case
case ‘C’ : led3 = 1; // LED P3.7 off
clear_remote(); // Clear command
break; // Out of from case
case ‘d’ : beep(); // Beep 1 time
clear_remote(); // Clear command
break; // Out of from case
case ‘D’ : beep(); // Beep 1 time
clear_remote(); // Clear command
break; // Out of from case
}
}
}
```

Приложение Б

```
/*-------------------------------------------------------------------------*/
// Program : Remote control robot
// Description: Robot controlled by ER-4 remote control Robot receive command
// : from remote control by serial communication at baud 1200 bps
// Filename : act1901.c
// C compiler : RIDE 51 V6.1
/*-------------------------------------------------------------------------*/
#include <C51ac2.h> // Header include register of T89C51AC2
#include <delay_robo51.h> // Delay time module function
#include <remote.h> // ER-4 Remote Control function
#include <sound_robo51.h> // Sound module function
#include <dc_motor.h> // DC motor module function
#include <lcd_robo51.h> // LCD module function
#define pow 200 // Define constant for power drive DC motor
void run_fd(int time_spin)
{
motor_fd(2,pow); // Motor channel 2 forward
motor_fd(1,pow); // Motor channel 1 forward
delay_ms(time_spin); // Delay time for robot drive forward
}
void run_bk(int time_spin)
{
motor_bk(2,pow); // Motor channel 2 backward
motor_bk(1,pow); // Motor channel 1 backward
delay_ms(time_spin); // Delay time for robot drive backward
}
void turn_left(int time_spin)
{
motor_fd(2,pow); // Motor channel 2 forward
motor_bk(1,pow); // Motor channel 1 backward
delay_ms(time_spin); // Delay time for robot spin turn left
}
void turn_right(int time_spin)
{
motor_bk(2,pow); // Motor channel 2 backward
motor_fd(1,pow); // Motor channel 1 forward
delay_ms(time_spin); // Delay time for robot spin turn right
}
void main()
{
beep(); // Beep 1 time
remote_init(); // Initial remote
while(1) // Infinite loop
{
switch(get_remote()) // Check command for receive
{
case ‘a’ : run_bk(100); // Drive robot backward when receive “a”
clear_remote(); // Clear command
break; // Out from case
case ‘A’ : run_bk(100); // Drive robot backward when receive “A”
clear_remote(); // Clear command
break; // Out from case
case ‘b’ : turn_right(100); // Turn right when receive “b”
motor_stop(all); // Robot stop
clear_remote(); // Clear command
break; // Out from case
case ‘B’ : turn_right(100); // Turn right when receive “B”
motor_stop(all); // Robot stop
```

```
clear_remote(); // Clear command
break; // Out from case
case ‘c’ : turn_left(100); // Turn right when receive “c”
motor_stop(all); // Robot stop
clear_remote(); // Clear command
break; // Out from case
case ‘C’ : turn_left(100); // Turn right when receive “C”
motor_stop(all); // Robot stop
clear_remote(); // Clear command
break; // Out from case
case ‘d’ : run_fd(100); // Turn right when receive “d”
clear_remote(); // Clear command
break; // Out from case
case ‘D’ : run_fd(100); // Turn right when receive “D”
clear_remote(); // Clear command
break; // Out from case
default : break; // Out from case
}
}
}
```

Задание по выполнению лабораторной работы:

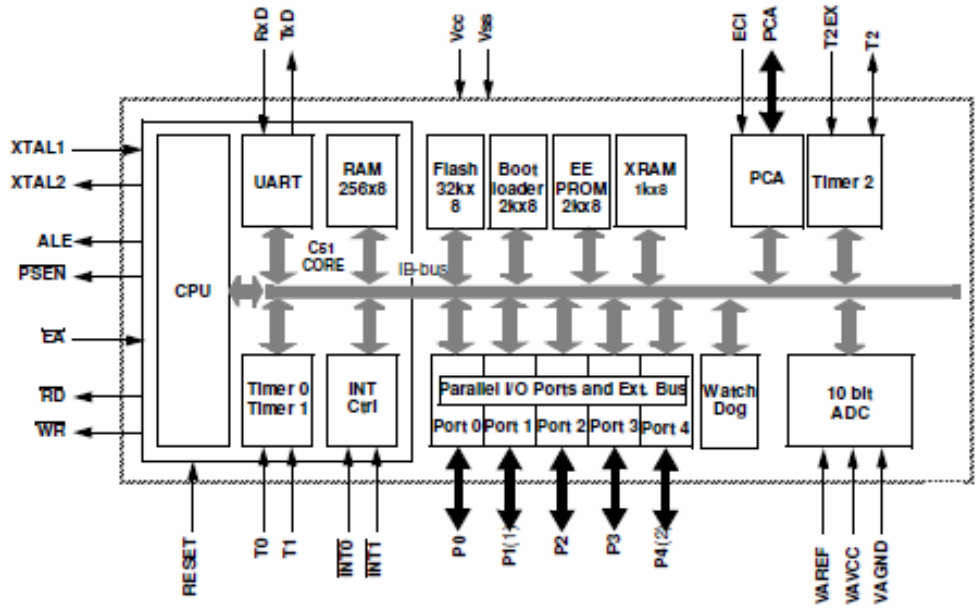
1) Изучение инфракрасного модуля очувствления робота.
2) Изучение инфракрасного пульта дистанционного управления роботом.
3) Составление тестовой программы для дистанционного управления IE-ROBO-51.
4) Запуск тестовой программы.
5) Дистанционное управление движением МС IE-ROBO-51.

Контрольные вопросы:

1) Коммуникационные способности МС.
2) Дистанционное управление движением промышленных роботов.
3) Современные тенденции развития МС.

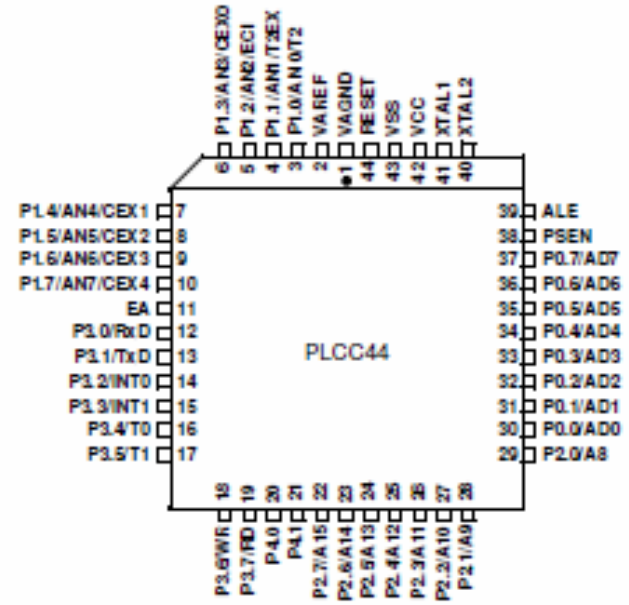
Дополнения

Block Diagram



Notes: 1. 8 analog Inputs/8 Digital I/O
2. 2-Bit I/O Port

Pin Configuration



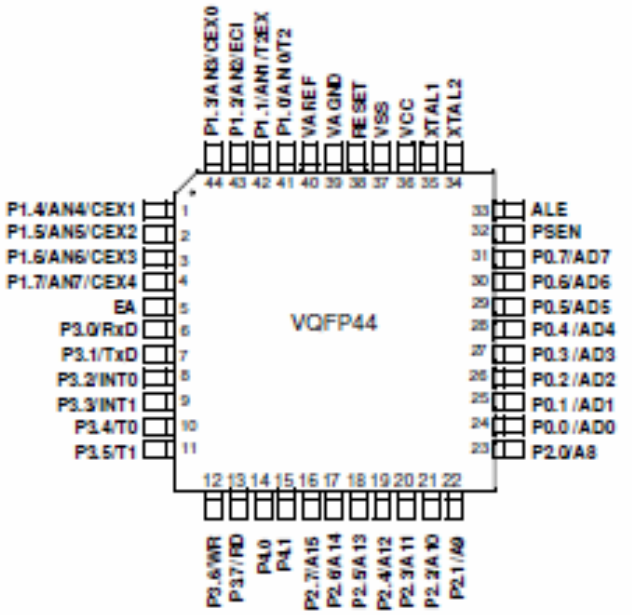


Table 1. Pin Description

Pin Name	Type	Description
VSS	GND	**Circuit ground**
VCC		**Supply Voltage**
VAREF		Reference Voltage for ADC
VAGND		Reference Ground for ADC
P0.0:7	I/O	**Port 0:** Is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in this state can be used as high-impedance inputs. Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pull-ups when emitting 1's. Port 0 also outputs the code Bytes during program validation. External pull-ups are required during program verification.
P1.0:7	I/O	**Port 1:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 1 pins can be used for digital input/output or as analog inputs for the Analog Digital Converter (ADC). Port 1 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 1 pins that are being pulled low externally will be the source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. Port 1 pins are assigned to be used as analog inputs via the ADCCF register (in this case the internal pull-ups are disconnected). As a secondary digital function, port 1 contains the Timer 2 external trigger and clock input; the PCA external clock input and the PCA module I/O. P1.0/AN0/T2 Analog input channel 0, External clock input for Timer/counter2. P1.1/AN1/T2EX Analog input channel 1, Trigger input for Timer/counter2. P1.2/AN2/ECI Analog input channel 2, PCA external clock input. P1.3/AN3/CEX0 Analog input channel 3, PCA module 0 Entry of input/PWM output. P1.4/AN4/CEX1 Analog input channel 4, PCA module 1 Entry of input/PWM output. P1.5/AN5/CEX2 Analog input channel 5, PCA module 2 Entry of input/PWM output. P1.6/AN6/CEX3 Analog input channel 6, PCA module 3 Entry of input/PWM output. P1.7/AN7/CEX4 Analog input channel 7, PCA module 4 Entry of input/PWM output. Port 1 receives the low-order address byte during EPROM programming and program verification. It can drive CMOS inputs without external pull-ups.
P2.0:7	I/O	**Port 2:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 2 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 2 pins that are being pulled low externally will be a source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. Port 2 emits the high-order address byte during accesses to the external Program Memory and during accesses to external Data Memory that uses 16-bit addresses (MOVX @DPTR). In this application, it uses strong internal pull-ups when emitting 1's. During accesses to external Data Memory that use 8 bit addresses (MOVX @RI), Port 2 transmits the contents of the P2 special function register. It also receives high-order addresses and control signals during program validation. It can drive CMOS inputs without external pull-ups.

Pin Name	Type	Description
P3.0:7	I/O	**Port 3:** Is an 8-bit bi-directional I/O port with internal pull-ups. Port 3 pins that have 1's written to them are pulled high by the internal pull-up transistors and can be used as inputs in this state. As inputs, Port 3 pins that are being pulled low externally will be a source of current (I_{IL}, see section "Electrical Characteristic") because of the internal pull-ups. The output latch corresponding to a secondary function must be programmed to one for that function to operate (except for TxD and $\overline{WR}$). The secondary functions are assigned to the pins of port 3 as follows: P3.0/RxD: Receiver data input (asynchronous) or data input/output (synchronous) of the serial interface P3.1/TxD: Transmitter data output (asynchronous) or clock output (synchronous) of the serial interface P3.2/$\overline{INT0}$: External interrupt 0 input/timer 0 gate control input P3.3/$\overline{INT1}$: External interrupt 1 input/timer 1 gate control input P3.4/T0: Timer 0 counter input P3.5/T1: Timer 1 counter input P3.6/$\overline{WR}$: External Data Memory write strobe; latches the data byte from port 0 into the external data memory P3.7/$\overline{RD}$: External Data Memory read strobe; Enables the external data memory. It can drive CMOS inputs without external pull-ups.
P4.0:1	I/O	**Port 4:** Is an 2-bit bi-directional I/O port with internal pull-ups. Port 4 pins that have 1's written to them are pulled high by the internal pull-ups and can be used as inputs in this state. As inputs, Port 4 pins that are being pulled low externally will be a source of current (IIL, on the datasheet) because of the internal pull-up transistor. P4.0 P4.1: It can drive CMOS inputs without external pull-ups.
RESET	I/O	**Reset:** A high level on this pin during two machine cycles while the oscillator is running resets the device. An internal pull-down resistor to VSS permits power-on reset using only an external capacitor to VCC.
ALE	O	**ALE:** An Address Latch Enable output for latching the low byte of the address during accesses to the external memory. The ALE is activated every 1/6 oscillator periods (1/3 in X2 mode) except during an external data memory access. When instructions are executed from an internal Flash ($\overline{EA}$ – 1), ALE generation can be disabled by the software.
PSEN	O	**$\overline{PSEN}$:** The Program Store Enable output is a control signal that enables the external program memory of the bus during external fetch operations. It is activated twice each machine cycle during fetches from the external program memory. However, when executing from of the external program memory two activations of PSEN are skipped during each access to the external Data memory. The PSEN is not activated for internal fetches.
EA	I	**$\overline{EA}$:** When External Access is held at the high level, instructions are fetched from the internal Flash when the program counter is less then 8000H. When held at the low level,A/T89C51AC2 fetches all instructions from the external program memory.
XTAL1	I	**XTAL1:** Input of the inverting oscillator amplifier and input of the internal clock generator circuits. To drive the device from an external clock source, XTAL1 should be driven, while XTAL2 is left unconnected. To operate above a frequency of 16 MHz, a duty cycle of 50% should be maintained.
XTAL2	O	**XTAL2:** Output from the inverting oscillator amplifier.

Table 2. Read-Modify-Write Instructions

Instruction	Description	Example
ANL	logical AND	ANL P1, A
ORL	logical OR	ORL P2, A
XRL	logical EX-OR	XRL P3, A
JBC	jump if bit = 1 and clear bit	JBC P1.1, LABEL
CPL	complement bit	CPL P3.0
INC	increment	INC P2
DEC	decrement	DEC P2
DJNZ	decrement and jump if not zero	DJNZ P3, LABEL
MOV Px.y, C	move carry bit to bit y of Port x	MOV P1.5, C
CLR Px.y	clear bit y of Port x	CLR P2.4
SET Px.y	set bit y of Port x	SET P3.3

SFR Mapping

The Special Function Registers (SFRs) of the A/T89C51AC2 fall into the following categories:

Table 3. C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ACC	E0h	Accumulator	–	–	–	–	–	–	–	–
B	F0h	B Register	–	–	–	–	–	–	–	–
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P
SP	81h	Stack Pointer	–	–	–	–	–	–	–	–
DPL	82h	Data Pointer Low byte LSB of DPTR	–	–	–	–	–	–	–	–
DPH	83h	Data Pointer High byte MSB of DPTR	–	–	–	–	–	–	–	–

Table 4. I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
P0	80h	Port 0	–	–	–	–	–	–	–	–
P1	90h	Port 1	–	–	–	–	–	–	–	–
P2	A0h	Port 2	–	–	–	–	–	–	–	–
P3	B0h	Port 3	–	–	–	–	–	–	–	–
P4	C0h	Port 4 (x2)	–	–	–	–	–	–	–	–

Table 5. Timers SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
TH0	8Ch	Timer/Counter 0 High byte	–	–	–	–	–	–	–	–
TL0	8Ah	Timer/Counter 0 Low byte	–	–	–	–	–	–	–	–
TH1	8Dh	Timer/Counter 1 High byte	–	–	–	–	–	–	–	–
TL1	8Bh	Timer/Counter 1 Low byte	–	–	–	–	–	–	–	–
TH2	CDh	Timer/Counter 2 High byte	–	–	–	–	–	–	–	–
TL2	CCh	Timer/Counter 2 Low byte	–	–	–	–	–	–	–	–
TCON	88h	Timer/Counter 0 and 1 control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
TMOD	89h	Timer/Counter 0 and 1 Modes	GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00

Table 5. Timers SFRs (Continued)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
T2CON	C8h	Timer/Counter 2 control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#
T2MOD	C9h	Timer/Counter 2 Mode	–	–	–	–	–	–	T2OE	DCEN
RCAP2H	CBh	Timer/Counter 2 Reload/Capture High byte	–	–	–	–	–	–	–	–
RCAP2L	CAh	Timer/Counter 2 Reload/Capture Low byte	–	–	–	–	–	–	–	–
WDTRST	A6h	Watchdog Timer Reset	–	–	–	–	–	–	–	–
WDTPRG	A7h	Watchdog Timer Program	–	–	–	–	–	S2	S1	S0

Table 6. Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
SCON	98h	Serial Control	FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI
SBUF	99h	Serial Data Buffer	–	–	–	–	–	–	–	–
SADEN	B9h	Slave Address Mask	–	–	–	–	–	–	–	–
SADDR	A9h	Slave Address	–	–	–	–	–	–	–	–

Table 7. PCA SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CCON	D8h	PCA Timer/Counter Control	CF	CR	–	CCF4	CCF3	CCF2	CCF1	CCF0
CMOD	D9h	PCA Timer/Counter Mode	CIDL	WDTE	–	–	–	CPS1	CPS0	ECF
CL	E9h	PCA Timer/Counter Low byte	–	–	–	–	–	–	–	–
CH	F9h	PCA Timer/Counter High byte	–	–	–	–	–	–	–	–
CCAPM0	DAh	PCA Timer/Counter Mode 0	–	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0
CCAPM1	DBh	PCA Timer/Counter Mode 1		ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1
CCAPM2	DCh	PCA Timer/Counter Mode 2		ECOM2	CAPP2	CAPN2	MAT2	TOG2	PWM2	ECCF2
CCAPM3	DDh	PCA Timer/Counter Mode 3		ECOM3	CAPP3	CAPN3	MAT3	TOG3	PWM3	ECCF3
CCAPM4	DEh	PCA Timer/Counter Mode 4		ECOM4	CAPP4	CAPN4	MAT4	TOG4	PWM4	ECCF4
CCAP0H	FAh	PCA Compare Capture Module 0 H	CCAP0H7	CCAP0H6	CCAP0H5	CCAP0H4	CCAP0H3	CCAP0H2	CCAP0H1	CCAP0H0
CCAP1H	FBh	PCA Compare Capture Module 1 H	CCAP1H7	CCAP1H6	CCAP1H5	CCAP1H4	CCAP1H3	CCAP1H2	CCAP1H1	CCAP1H0
CCAP2H	FCh	PCA Compare Capture Module 2 H	CCAP2H7	CCAP2H6	CCAP2H5	CCAP2H4	CCAP2H3	CCAP2H2	CCAP2H1	CCAP2H0
CCAP3H	FDh	PCA Compare Capture Module 3 H	CCAP3H7	CCAP3H6	CCAP3H5	CCAP3H4	CCAP3H3	CCAP3H2	CCAP3H1	CCAP3H0
CCAP4H	FEh	PCA Compare Capture Module 4 H	CCAP4H7	CCAP4H6	CCAP4H5	CCAP4H4	CCAP4H3	CCAP4H2	CCAP4H1	CCAP4H0

Table 7. PCA SFRs (Continued)

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
CCAP0L	EAh	PCA Compare Capture Module 0 L	CCAP0L7	CCAP0L6	CCAP0L5	CCAP0L4	CCAP0L3	CCAP0L2	CCAP0L1	CCAP0L0
CCAP1L	EBh	PCA Compare Capture Module 1 L	CCAP1L7	CCAP1L6	CCAP1L5	CCAP1L4	CCAP1L3	CCAP1L2	CCAP1L1	CCAP1L0
CCAP2L	ECh	PCA Compare Capture Module 2 L	CCAP2L7	CCAP2L6	CCAP2L5	CCAP2L4	CCAP2L3	CCAP2L2	CCAP2L1	CCAP2L0
CCAP3L	EDh	PCA Compare Capture Module 3 L	CCAP3L7	CCAP3L6	CCAP3L5	CCAP3L4	CCAP3L3	CCAP3L2	CCAP3L1	CCAP3L0
CCAP4L	EEh	PCA Compare Capture Module 4 L	CCAP4L7	CCAP4L6	CCAP4L5	CCAP4L4	CCAP4L3	CCAP4L2	CCAP4L1	CCAP4L0

Table 8. Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
IEN0	A8h	Interrupt Enable Control 0	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
IEN1	E8h	Interrupt Enable Control 1	–	–	–	–	–	–	EADC	–
IPL0	B8h	Interrupt Priority Control Low 0	–	PPC	PT2	PS	PT1	PX1	PT0	PX0
IPH0	B7h	Interrupt Priority Control High 0	–	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H
IPL1	F8h	Interrupt Priority Control Low 1	–	–	–	–	–	–	PADCL	–
IPH1	F7h	Interrupt Priority Control High1	–	–	–	–	–	–	PADCH	–

Table 9. ADC SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
ADCON	F3h	ADC Control	–	PSIDLE	ADEN	ADEOC	ADSST	SCH2	SCH1	SCH0
ADCF	F6h	ADC Configuration	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
ADCLK	F2h	ADC Clock	–	–	–	PRS4	PRS3	PRS2	PRS1	PRS0
ADDH	F5h	ADC Data High byte	ADAT9	ADAT8	ADAT7	ADAT6	ADAT5	ADAT4	ADAT3	ADAT2
ADDL	F4h	ADC Data Low byte	–	–	–	–	–	–	ADAT1	ADAT0

Table 10. Other SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0
PCON	87h	Power Control	SMOD1	SMOD0	–	POF	GF1	GF0	PD	IDL
AUXR	8Eh	Auxiliary Register 0	–	–	M0	–	XRS1	XRS2	EXTRAM	A0
AUXR1	A2h	Auxiliary Register 1	–	–	ENBOOT	–	GF3	0	–	DPS
CKCON	8Fh	Clock Control	–	WDX2	PCAX2	SIX2	T2X2	T1X2	T0X2	X2
FCON	D1h	Flash Control	FPL3	FPL2	FPL1	FPL0	FPS	FMOD1	FMOD0	FBUSY
EECON	D2h	EEPROM Contol	EEPL3	EEPL2	EEPL1	EEPL0	–	–	EEE	EEBUSY

Table 11. SFR Mapping

	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h	IPL1 xxxx xx0x	CH 0000 0000	CCAP0H 0000 0000	CCAP1H 0000 0000	CCAP2H 0000 0000	CCAP3H 0000 0000	CCAP4H 0000 0000		FFh
F0h	B 0000 0000		ADCLK xxx0 0000	ADCON x000 0000	ADDL 0000 0000	ADDH 0000 0000	ADCF 0000 0000	IPH1 xxxx xx0x	F7h
E8h	IEN1 xxxx xx0x	CL 0000 0000	CCAP0L 0000 0000	CCAP1L 0000 0000	CCAP2L 0000 0000	CCAP3L 0000 0000	CCAP4L 0000 0000		EFh
E0h	ACC 0000 0000								E7h
D8h	CCON 00x0 0000	CMOD 00xx x000	CCAPM0 x000 0000	CCAPM1 x000 0000	CCAPM2 x000 0000	CCAPM3 x000 0000	CCAPM4 x000 0000		DFh
D0h	PSW 0000 0000	FCON 0000 0000	EECON xxxx xx00						D7h
C8h	T2CON 0000 0000	T2MOD xxxx xx00	RCAP2L 0000 0000	RCAP2H 0000 0000	TL2 0000 0000	TH2 0000 0000			CFh
C0h	P4 xxxx xx11								C7h
B8h	IPL0 x000 0000	SADEN 0000 0000							BFh
B0h	P3 1111 1111							IPH0 x000 0000	B7h
A8h	IEN0 0000 0000	SADDR 0000 0000							AFh
A0h	P2 1111 1111		AUXR1 xxxx 00x0				WDTRST 1111 1111	WDTPRG xxxx x000	A7h
98h	SCON 0000 0000	SBUF 0000 0000							9Fh
90h	P1 1111 1111								97h
88h	TCON 0000 0000	TMOD 0000 0000	TL0 0000 0000	TL1 0000 0000	TH0 0000 0000	TH1 0000 0000	AUXR x00x 1100	CKCON 0000 0000	8Fh
80h	P0 1111 1111	SP 0000 0111	DPL 0000 0000	DPH 0000 0000				PCON 00x1 0000	87h
	0/8[1]	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

Reserved

Note: 1. These registers are bit-addressable.
Sixteen addresses in the SFR space are both byte-addressable and bit-addressable. The bit-addressable SFR's are those whose address ends in 0 and 8. The bit addresses, in this area, are 0x80 through to 0xFF.

Список литературы

1) Robotics experiment with MCS-51 microcontroller based-on Robo-51 robot kit 2nd Edition (C) Innovative Experiment Co.,Ltd.

2) Atmel Corporation 2008 4127H–8051–02/08

3) Мехатроника: Пер с япон. / Исии Х., Иноуэ Х., Симояма И. и др. — М.: Мир, 1988. — С. 318. — ISBN 5-03-000059-3

4) Введение в мехатронику: В 2-х кн. Учебное пособие / А. К. Тугенгольд, И. В. Богуславский, Е. А. Лукьянов, В. В. Мартынов, В. А. Герасимов, Ю. Б. Ивацевич, Н. Ф. Карнаухов, В. А. Череватенко. Под ред. А. К. Тугенгольда. — Ростов н/Д: Издательский центр ДГТУ, 2004. — ISBN 5-7890-0294-3

5) Карнаухов Н. Ф. Электромеханические и мехатронные системы. — Ростов н/Д: Феникс, 2006. — 320 с. — (Высшее образование) — 3000 экз. — ISBN 5-222-08228-8

6) Егоров О. Д., Подураев Ю. В. Конструирование мехатронных модулей. — М.: Издательство МГТУ «Станкин», 2004. — С. 368.

7) Подураев Ю. В. Мехатроника. Основы, методы, применение. — 2-е изд., перераб и доп. — М.: Машиностроение, 2007. — С. 256. — ISBN 978-5-217-03388-1

8) Ю. В. Подураев «Основы мехатроники» Учебное пособие. Москва.- 2000г. 104 с.

9) А.В.Башарин, В.А.Новиков, Г.Г. Соколовский. Управление электроприводами (Для студентов ВУЗов). Ленинград, Энергоиздат, Ленинградское отделение,1982г.

10) Ю.А.Борцов, Г.Г.Соколовский. Автоматизированный электропривод с упругими связями.Санкт-Петербург, Энергоатомиздат, 1992г.

11) Е.Бенькович, Ю.Колесов, Ю.Сениченков. Практическое моделирование динамических систем.БХВ –Петербург, 2002г.

12) Ю.М.Беленький, А.Г. Микеров. Выбор и программирование параметров бесконтактного моментного привода. Л.: ЛДНТП, 1990.

13) С.Г. Герман-Галкин. Компьютерное моделирование полупроводниковых систем. -Санкт-Петербург. "КОРОНА принт", 2001.

14) Т.А. Глазенко, Р.Б. Гончаренко. Полупроводниковые преобразователи частоты в электроприводах.-Л.: Энергия, 1969.

15) Т.А. Глазенко. Полупроводниковые преобразователи в электроприводах постоянного тока.-Л.: Энергия, 1973.

16) Забродин Ю. С. Промышленная электроника.- М.: Высшая школа, 1982.

17) Н.Ф.Ильинский. Основы электропривода. Учебное пособие для ВУЗов. Москва, Изд.МЭИ, 2003г.

18) Системы подчиненного регулирования электроприводов переменного тока с вентильными преобразователями. О.В. Слежановский, Л.Х. Дацковский, И.С. Кузнецов, Е.Д. Лебедев, Л.М. Тарасенко.Москва, Энергоатомиздат, 1983.

19) А.А.Усольцев. Векторное управление асинхронными двигателями.Учебное посоие по дисциплинам электромеханического цикла. Санкт-Петербург, 2002г.

20) .Б.И.Фигаро, Л.Б Павлячек. Теория электропривода. Минск, ЗАО Техноперспектива , 2004г.

21) Ч. Филипс, Р. Харбор. Системы управления с обратной связью. Москва. Лаборатория базовых знаний. 2001.

22) Anatolij Afonin, Piotr Szymczak Mechatronika Politechnika Szczecińska, Wydzial Elektryczny, Szczecin, 2001.

23) Gil A. Podstawy elektroniki i energoelektroniki. Cz.2- Gdynia 1998. s. 155.

Szczęsny R. Komputerowa symulacja układów energoelektronicznych. Wyd. Politechniki Gdańskej, 1999.