

# Азбука ARMатурщика

ОСНОВЫ БЫТОВОЙ АВТОМАТИКИ, ЭЛЕКТРОНИКИ И СИСТЕМ УПРАВЛЕНИЯ

© Консорциум хоббитов России  
HackSpace «Чебураторный завод»  
Дмитрий Понятов <dponyatov@gmail.com>  
группа ruOpenWrt  
Bill Collis (II)  
Andreas Fester (IV)  
Joe Martin, Craig Libuse (12)

# Оглавление

I	О книге	11
II	Введение в практическую электронику	14
	An Introduction to Practical Electronics, Microcontrollers and Software Design © Bill Collis . . . . .	15
1	Введение в практическую электронику	16
2	Ваше обучение по специальности «Технология»	18
2.1	Цели обучения технологиям Ново-Зеландской программы . . . . .	18
2.2	Ключевые компетенции Ново-Зеландской программы . . . . .	19
3	Вводная электронная схема	22
3.1	Где купить комплектующие? . . . . .	22
3.2	Макетная плата: breadboard . . . . .	23
3.3	Простейшая схема . . . . .	24
3.4	Ток и напряжение, сечение проводника, плотность тока . . . . .	25
3.5	Проводники и изоляторы, сопротивление и проводимость . . . . .	27

3.6	Диэлектрическая и тепловая прочность изоляции . . . . .	29
3.7	Тепловое действие тока. Мощность . . . . .	30
3.8	Масштабные множители . . . . .	31
3.9	Использование мультиметра . . . . .	32
3.10	Определение сопротивления резистора по цветовому коду . . . . .	35
 <b>III САПР электронных устройств KiCAD</b>		<b>37</b>
4	<b>eeschema</b> : редактор электрических схем	38
 <b>IV Расчет схем и моделирование в ngSPICE</b>		<b>39</b>
5	Доступные SPICE-пакеты	41
5.1	Установка <b>ngSPICE</b> под Windows . . . . .	42
5.2	Установка <b>LT-SPICE</b> (только Windows) . . . . .	42
5.3	Установка <b>ngSPICE</b> под Linux . . . . .	43
6	Пошаговый пример использования	44
6.1	Рисуем схему в <b>KiCAD</b> . . . . .	44
6.2	Создание списка цепей . . . . .	46
6.3	Запуск симуляции . . . . .	48
6.4	Просмотр результата расчета . . . . .	50
6.5	Расчет АЧХ по переменному току (АС симуляция) . . . . .	54
6.6	Симуляция полноволнового выпрямителя . . . . .	58






<b>7</b>	<b>Настройка KiCAD для SPICE-моделирования</b>	<b>61</b>
7.1	Библиотеки компонентов со SPICE-элементами . . . . .	61
7.2	Настройка проекта . . . . .	62
7.3	Как это работает . . . . .	62
<b>V</b>	<b>Разработка конструкции в САПР FreeCAD</b>	<b>64</b>
<b>VI</b>	<b>Установка под Windows</b>	<b>67</b>
<b>VII</b>	<b>Инструменты и электронное оборудование</b>	<b>69</b>
<b>8</b>	<b>Радиомонтажный инструмент</b>	<b>70</b>
8.1	Pro'sKit . . . . .	70
8.2	Инструмент до 1000 В . . . . .	75
8.3	Хранение . . . . .	76
8.4	Радиомонтаж . . . . .	77
8.5	Прочие . . . . .	81
<b>9</b>	<b>Паяльное оборудование</b>	<b>82</b>
9.1	Паяльник . . . . .	82
9.2	Паяльная станция . . . . .	84
<b>10</b>	<b>Измерительное оборудование</b>	<b>87</b>
10.1	Мультиметр . . . . .	87
10.1.1	Mastech M838 . . . . .	88

10.1.2	Mastech M300 . . . . .	89
10.1.3	Mastech M320 . . . . .	90
10.2	Осциллограф . . . . .	91
10.3	Логический анализатор . . . . .	91
10.4	Генератор сигналов . . . . .	91
10.5	Рыльцемер RLC . . . . .	91
<b>11</b>	<b>Электроинструмент</b>	<b>92</b>
11.1	Дрель . . . . .	93
11.2	Лобзик . . . . .	96
11.3	Жвигатель . . . . .	97
<b>VIII</b>	<b>Станочное оборудование</b>	<b>100</b>
<b>12</b>	<b>Настольные станки</b>	<b>103</b>
<b>13</b>	<b>Самодельная оснастка</b>	<b>104</b>
<b>14</b>	<b>Промышленные станки</b>	<b>105</b>
14.1	1A616: станок токарно-винторезный . . . . .	106
14.1.1	Назначение и области применения . . . . .	107
14.1.2	Распаковка и транспортировка . . . . .	107
14.1.3	Фундамент станка, монтаж и установка . . . . .	107
14.1.4	Подготовка станка к первоначальному пуску . . . . .	107
14.1.5	Паспортные данные . . . . .	107
14.1.6	Описание основных узлов . . . . .	107
14.1.7	Смазка . . . . .	107

14.1.8 Первоначальный пуск . . . . .	107
14.1.9 Указания по технике безопасности . . . . .	107
14.1.10 Настройка . . . . .	107
14.1.11 Регулирование . . . . .	107
14.1.12 Ведомость комплектации . . . . .	107

## IX Разработка ПО для встраиваемых систем 108

### 15 IDE 109

15.1  ECLIPSE . . . . .	112
15.1.1 Установка  ECLIPSE под  Windows . . . . .	112
15.1.2 Установка  ECLIPSE под Linux . . . . .	112
15.1.3 Установка CDT . . . . .	113
15.1.4 Установка PyDev . . . . .	114
15.1.5 Установка TeXlipse . . . . .	114
15.1.6 Редактирование файлов в формате XML и производных . . . . .	115
15.1.7 Проверка орфографии . . . . .	115
15.2 Code::Blocks . . . . .	118
15.3 (g)Vim . . . . .	118
15.3.1 Установка под  Windows . . . . .	118
15.3.2 Выход из (g)Vim . . . . .	120
15.3.3 Выход с автосохранением . . . . .	121
15.3.4 Переход в режим редактирования . . . . .	121
15.3.5 Переход в режим команд . . . . .	121
15.3.6 Запись редактируемого файла . . . . .	121
15.3.7 Перезагрузка файла . . . . .	122

15.3.8	Отмена последних изменений (undo)	122
16	<b>Make: управление сборкой проектов</b>	<b>123</b>
17	<b>VCS: системы контроля версий</b>	<b>124</b>
17.1	CVS	124
17.2	Subversion	124
17.3	Git	124
17.3.1	GitHub	124
18	<b>Вспомогательные скрипты на языке Python</b>	<b>125</b>
19	<b>Основы Си и <math>C_+^+</math></b>	<b>126</b>
19.1	Установка MinGW (win32)	126
19.2	Особенности $C_+^+$ в embedded	126
19.3	Сборка кросс-компилятора GNU toolchain	126
20	<b>Лексический и синтаксический анализ</b>	<b>127</b>
20.1	Лексер и лексический анализ, утилита flex	128
20.2	Генератор синтаксических анализаторов bison	134
20.3	Семантический анализ	134
20.4	Оптимизация	135
20.5	Кодогенерация	135
20.6	Транслятор Паскаля	135
20.7	LLVM и разработка собственных компиляторов	135

<b>X</b>	<b>Микроконтроллеры Cortex-Mx</b>	<b>136</b>
<b>21</b>	<b>Отладочные платы</b>	<b>137</b>
21.1	STM32DISCOVERY /Cortex-M3 STM32F103/ . . . . .	137
21.2	STM32F4DISCOVERY /Cortex-M4 STM32F407/ . . . . .	137
21.3	Maple Mini /Cortex-M3 STM32F100/ . . . . .	137
<b>XI</b>	<b>Встраиваемый emLinux</b>	<b>138</b>
<b>22</b>	<b>Загрузчик syslinux</b>	<b>140</b>
22.1	Закачка . . . . .	140
22.2	Установка под Windows на флешку . . . . .	141
22.3	syslinux.cfg . . . . .	142
<b>23</b>	<b>azLinux</b>	<b>145</b>
23.1	Требования к системе сборки (BUILD-хост) . . . . .	147
23.2	Понятие пакет . . . . .	148
23.3	Клонирование проекта azLinux . . . . .	150
23.4	Общий порядок сборки . . . . .	150
23.5	Фиксация переменных . . . . .	152
23.6	dirs: Создание дерева каталогов . . . . .	152
23.7	gz: Загрузка архивов исходников . . . . .	155
23.8	APP: Приложение . . . . .	157
23.9	HW: Поддерживаемое железо . . . . .	158
23.10	386 . . . . .	159
23.10.1	qemu386: эмулятор QEMU . . . . .	159



23.10.2	eeepc701: ASUS Eee PC 701	159
23.10.3	gac1037: Gigabyte GA-C1037UN-EU rev.2	160
23.11	ARM	162
23.11.1	qemuARM: эмулятор QEMU	162
23.11.2	cubie1: Cubie Board v.1	163
23.11.3	rpi: Raspberry Pi model B	163
23.12	MIPS	163
23.12.1	qemuMIPS: эмулятор QEMU	163
23.12.2	mr3020: роутер MR3020	163
23.12.3	vocore: VoCore	163
23.12.4	bswift: BlackSwift	163
23.13	CPU: Конфигурации процессоров	164
23.13.1	i386	164
23.13.2	ARM	165
23.13.3	MIPS	165
23.14	Пакеты	165
23.14.1	versions.mk: Версии пакетов	165
23.14.2	tc: кросс-компилятор	165
23.14.3	core: ядро	166
23.14.4	boot: загрузчики	166
23.14.5	libs: библиотеки	166
23.14.6	tc: сборка кросс-компилятора	167
23.14.7	binutils: ассемблер, линкер и утилиты	170
23.14.8	cclibs: библиотеки для сборки gcc	171
23.14.9	gcc0: сборка минимального кросс-компилятора Си	171
23.14.10	gcc: пересборка полного кросс-компилятора Си/C <sup>+</sup>	172
23.14.11	core: сборка основной системы	172

23.14.12	kernel: ядро Linux	172
23.14.13	libc: библиотека uClibc	178
23.14.14	gcc: пересборка полного gcc	184
23.14.15	busybox: набор утилит busybox	184
23.14.16	libs: сборка библиотек \${LIBS}	185
23.14.17	apps: сборка прикладных пакетов \${APPS}	185
23.14.18	user: сборка пользовательского кода	185
23.14.19	root: формирование корневой файловой системы	185
23.14.20	boot: сборка загрузчика syslinux/grub/uboot	188
23.14.21	syslinux	188
23.14.22	grub	188
23.14.23	uboot	188
23.14.24	emu: запуск собранной системы в эмуляторе	188
23.15	netboot: Сетевая загрузка	190
23.16	Прошивка на устройство	190
23.17	RT-патч	190
24	BuildRoot	191
25	Особенности OpenWrt	192
XII	Символьная и численная математика	193
26	Общие сведения о компьютерной математике	195
27	Пакет Maxima	198
27.1	Установка Maxima под Windows	198

27.2 Калькулятор . . . . .	199
<b>XIII Подготовка технической документации</b>	<b>200</b>
28 Верстка в $\text{\LaTeX}$	201
29 Оформление листингов	202
30 Подготовка иллюстраций	203
30.1 GIMP . . . . .	203
30.2 Inkscape . . . . .	203
30.3 Graphviz . . . . .	203
<b>XIV Примерные учебные планы</b>	<b>204</b>
31 Блондинко	205
32 Школотрон	206
33 Студень	207
34 Технический специалист	208

XV	Куча	209
----	------	-----

Предметный указатель	210
----------------------	-----

Часть I

О книге

Это учебное пособие было создано для интересующихся любительской электроникой, самодельными цифровыми системами управления (Arduino, устройствами на микроконтроллерах и т.п.), и программистов-любителей. В связи с полной деградацией системы образования пособие также рекомендуется для применения при обучении в ВУЗах по специализациям, связанным с применением цифровой электроники и компьютерной техники.

Большой упор был сделан на использование открытого некоммерческого программного обеспечения, для удешевления учебного процесса, уменьшения себестоимости ваших проектов<sup>1</sup>, и стимулирования вашего участия в развитии этих программных пакетов.

Книга очень объемна и разнообразна по материалу, и построена как справочник с группировкой материала по тематике. Для тех, кто только начинает, в разделе XIV расписаны **пошаговые учебные планы** с точки зрения параллельного изучения нескольких предметов с постепенным усложнением<sup>2</sup>. Как известно, главная часть любого обучения — практическая. Особое внимание уделено набору лабораторных работ.

В качестве видеоматериала были использованы видеоуроки физики

© Ерюткин Е.С., учитель физики высшей категории ГБОУ СОШ №1360, г.Москва

Мы признательны Bill Collis за разрешение использовать материалы его книги «An Introduction to Practical Electronics, Microcontrollers and Software Design» [?] в русскоязычном варианте «Азбуки» (II), и конечно он вполне заслуженно включен в основные соавторы этой книги.

Так как для работы в области электроники необходимо владение технологиями изготовления конструктива, в книгу включен соответствующий раздел. Эти книги рекомендуются популярным поставщиком хоббийных настольных микро-станков Sherline Products. Так как от владельцев авторских прав не получено разрешение на полный официальный перевод, для этих книг сделан только перевод-подстрочник, который поможет вам читать оригинал:

---

<sup>1</sup> вряд ли у вас окажется лишняя пачка килобаксов на покупку пары коммерческих САПР, по крайней мере пока ваш стартап не взлетит в Top\$100K

<sup>2</sup> как это происходит при традиционном offline обучении

- Joe Martin, Craig Libuse **Tabletop Machining** [?] (12)
- Doug Briney **Home Machinists Handbook** [?] (??)

Отечественных книг по использованию маленьких “часовых” и настольных станков просто не существует, хотя они и выпускались серийно. Исключение — книга Евгений Васильев **Маленькие станки** [?], по согласованию в автором включена как отдельный раздел (??).

Лицензия на эту книгу пока не выбрана, так что она пока просто пишется в духе OpenSource: любой может использовать ее часть, изменять или дополнять, до тех пор, пока не накладываются какие-либо административные, финансовые или юридические ограничения на распространение и развитие оригинальной версии или ее открытых форков: <https://github.com/ponyatov/Azbuka>

Приглашаем всех желающих участвовать в развитии этого учебного пособия на форум ruOpenWrt и в группу <http://vk.com/samarahackerspace>, нам нужна обратная связь по качеству материала, результаты тестирования на вас или ваших студентах, дополнения и замечания.

## Часть II

# Введение в практическую электронику



Эта часть основана на книге:

**An Introduction to Practical Electronics, Microcontrollers and Software Design**

Second Edition, 01 May-2014

© Bill Collis

[www.techideas.co.nz](http://www.techideas.co.nz)

Мы признательны автору за разрешение использовать материалы его книги в русскоязычном варианте «Азбуки», и конечно он вполне заслуженно включен в основные соавторы этой книги.

We are grateful to the author for permission to use materials of his book in the russian version of «Azbuka», and of course he was deservedly included in the main co-authors of this book.

From: Bill Collis <Bill.Collis@.....nz>

Date: 2014-11-24 0:53 GMT+04:00

Subject: Electronis Book

To: "dponyatov@gmail.com" <dponyatov@gmail.com>

Hi Dmitry

thanks for your email.

I am looking at the future of the book myself and thinking I will open source it. If you will only be in using it in Russian language then that is ok and you need to reference the original book.

Thanks

Bill

# Глава 1

## Введение в практическую электронику

Эта книга ©<sup>1</sup> имеет следующий ряд основных направлений:

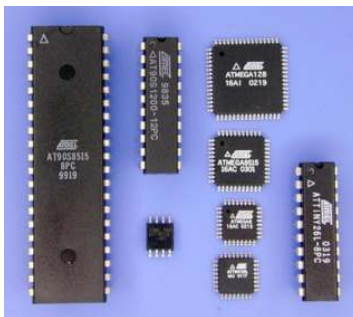
- Распознавание электронных компонентов и их правильное использование
- Нарботка цельного набора компетенций по основам электроники
- Использование макетных плат
- Навыки ручной пайки
- Использование закона Ома для выбора токоограничивающих резисторов
- Делитель напряжения
- Использование EDA CAD<sup>2</sup> для разработки и подготовки производства печатных плат

---

<sup>1</sup> оригинал: [?] B.Collis The Introduction to Practical Electronics. ...

<sup>2</sup> [E]lectronic [D]esign [A]utomation, САПР автоматизации проектирования электроники

- Программирование микроконтроллеров и их сопряжение с внешними устройствами
- Использование транзистора в режиме ключа
- Теория источников питания
- Принципы и схемы электропривода
- Навыки отладки схем, их тестирования и испытаний
- Следование принципам обучения через практику
- Безопасные приемы работы



# Глава 2

## Ваше обучение по специальности «Технология»

### 2.1 Цели обучения технологиям Ново-Зеландской программы

- Технологическая практика

- **Чоткость:** разработка ясных описаний для ваших технологических проектов.
- **Планирование:** думать прежде чем делать, и использовать во время работы документацию: блок-схемы, принципиальные схемы, чертежи разводки плат, диаграммы и эскизы.
- **Наработка навыков:** сборка, отладка и тестирование электронных схем, проектирование и изготовление печатных плат, написание программ для микроконтроллеров.

- Технологические знания

- **Моделирование:** прежде чем строить готовое электронное устройство, сначала важно понять как оно работает путем моделирования и/или макетирования аппаратного и программного обеспечения.
- **Технологические продукты:** знания о компонентах и их характеристиках.
- **Технологические системы:** электронное устройство является более, чем набором компонентов, это функционирующая система с входами, выходами и контролирующим процессом.

- **Природа технологии**

- **Значение технологических достижений:** знания об электронных компонентах, особенно микроконтроллерах, как основе современных технологий.
- **Роль технологии в обществе:** электронные устройства в настоящее время играют центральную роль в инфраструктуре нашего современного общества; подчинили ли они нас себе, как они изменили нашу жизнь?

## 2.2 Ключевые компетенции Ново-Зеландской программы

- **Знания:** для меня предметом технологии является все что относится к знанию. Моя цель: заставить студентов понимать технологии, заложенные в электронные устройства. Для достижения этого понимания студенты должны активно учиться<sup>1</sup> в работе на самом раннем этапе, чтобы они могли построить собственное понимание предмета и пойти дальше, чтобы стать хорошими решателями проблем. В начале обучения электронике это требует от студентов восприимчивости к инструкциям, которые им дают, и поиск ясности, когда они не понимают их.

Для этого на занятиях рассматриваются много новых и различных элементов знаний, и студентам выдаются задания на решение проблем, чтобы помочь им мыслить логически. Копирование чужого ответа

---

<sup>1</sup> в оригинале **enage**, англо-калька с [себуанского](#), NZ

наказывается, но приветствуется совместная работа. В основе обучения лежит построение правильных концептуальных моделей и анализ в контексте "большой картины".

- **Взаимодействие:** работа в парах и группах, это важно как в классе, так и в любой другой ситуации в жизни; мы все должны договариваться и разделять ресурсы и оборудование с другими людьми; поэтому крайне важно активное общение и помощь друг другу.
- **Использование языка символов и текстов:** сердцем нашего предмета является язык, который мы используем для обмена информацией в электронных схемах, планах, алгоритмах и синтаксисе компьютерных языков программирования; так что способность распознавать и правильно использовать символы и диаграммы для работы, которую мы делаем, имеет критическое значение.
- **Самоконтроль:** студенты принимают на себя личную ответственность за собственное обучение; они принимают вызов, надеясь найти ответы в книгах или найти учителя, способного объяснить им, что делать. Это значит, что студенты должны взаимодействовать с рабочим материалом.  
  
Иногда ответы приходят легко, иногда нет; часто наша тема требует много проб и ошибок (в основном ошибок). Студенты должны знать, что у них будут трудные времена, пока не будет изучена большая часть. И не сдаваться в поиске понимания.
- **Участие и содействие:** мы живем в мире, который невероятно зависит от технологии, особенно электроники; студенты должны развивать осознание важности этой области человеческого творчества в нашей повседневной жизни, и понимать, что наши проекты имеют и социальную функцию, а не только техническую.



# Глава 3

## Вводная электронная схема

### 3.1 Где купить комплектующие?

В Новой Зеландии есть некоторое количество отличных поставщиков компонентов с разумными ценами, включающих [www.surplustronics.co.nz](http://www.surplustronics.co.nz), и [www.activecomponents.com](http://www.activecomponents.com). Зарубежные поставщики, которых я использую, включают [www.digikey.co.nz](http://www.digikey.co.nz), [www.sparkfun.com](http://www.sparkfun.com), [ebay.com](http://ebay.com) и [aliexpress.com](http://aliexpress.com).

Для России можно отметить сеть магазинов “Болтмастер”<sup>1</sup>.

Для модулей пока что доступна поставка из Китая по почте: AliExpress с оплатой с виртуальной карты VISA платежной системы QIWI. Доставка занимает от 2х недель до 2х месяцев. Удобно покупать модули в виде микросхем, уже запаянных с обвязкой на кусок текстолита (**breakout board**): Arduino Mini, Maple Mini, датчики, контроллеры двигателей. Также интересны наборы (магазины) элементов: пачки резисторов, конденсаторов и т.п. в выводном исполнении, по несколько десятков номиналов.

---

<sup>1</sup> Самара



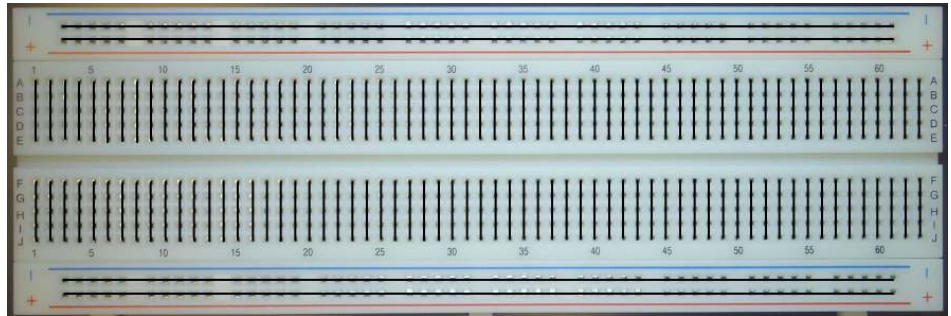
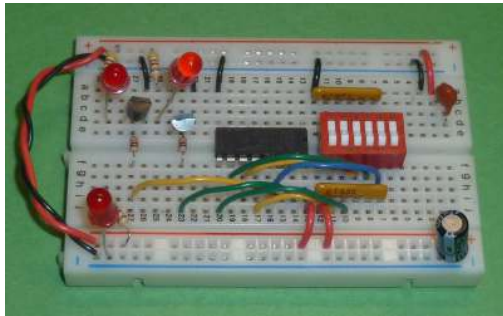
## 3.2 Макетная плата: breadboard



<https://www.youtube.com/watch?v=vQdUSE1auz8>

<https://www.youtube.com/watch?v=k9jcHB9tWko>

[https://www.youtube.com/watch?v=2wvn8\\_23phE](https://www.youtube.com/watch?v=2wvn8_23phE)



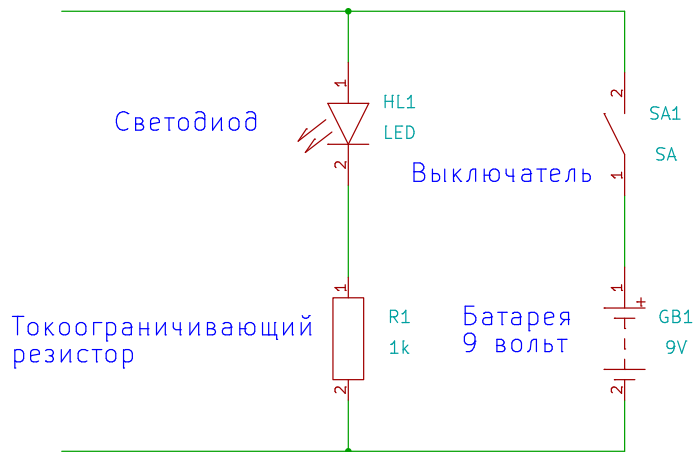
**Breadboard** ([б]еспаяечная [м]акетная [п]лата, БМП, “вафля”) — пластмассовый блок с отверстиями и металлическими полосковыми зажимами, создающими соединения между **элементами схемы**. Отверстия расположены так, что **компоненты** и отрезки провода могут быть соединены вместе формируя схему, без использования паяльника. Верхние и нижние ряды, как правило, используются для **шин питания**, красный сверху для плюса, и внизу черный/синий для минуса (**общий провод**, или “земля”). На длинных вафлях шины питания поделены на отдельные сегменты, и требуют соединения короткими перемычками.

## 3.3 Простейшая схема

Эта схема может быть собрана вот так 3.3, обратите внимание, что **светодиод** должен находиться в правильном положении. Если у вас есть светодиод и **резистор**, соединенные в **замкнутый контур**, светодиод должен загореться.

Принципиальная схема

Компоновка



Светодиоду требуется для включения около  $2\text{ В}^2$ , батарея на 9 В, так что с напряжением все нормально. Но если вы подключите светодиод напрямую к батарее, он сгорит! Для светодиодов главный рабочий параметр — допустимый рабочий ток, обычно он не превышает  $10..20\text{ mA}^3$ . Так что  $1\text{ k}^4$  резистор используется для ограничения тока через светодиод.

<sup>2</sup> [В]ольт

<sup>3</sup> 1 [м]иллиАмпер = 0.001 [А]мпер

<sup>4</sup> 1 [К]илоОм = 1000 Ом

Если вы отключите любой провод в схеме, она перестает работать, схема должна быть завершенной, чтобы электроны могли течь по проводникам из **источника питания**.

## 3.4 Ток и напряжение, сечение проводника, плотность тока



<https://www.youtube.com/watch?v=Dq4fSp6wz-o>

1. [https://www.youtube.com/watch?v=3ZeveDL1\\_bg](https://www.youtube.com/watch?v=3ZeveDL1_bg) Электрический ток Источники тока
2. <https://www.youtube.com/watch?v=mt86jYUjPFI> Ток в металлах Действия электрического тока
3. <https://www.youtube.com/watch?v=42CEi94hGgA> Электрический ток. Сила тока
4. <https://www.youtube.com/watch?v=SNP05e9wZWU> Амперметр Измерение силы тока
5. <https://www.youtube.com/watch?v=Yt91aLAwp68> Электрическое напряжение

В предыдущем разделе были использованы два важных понятия электроники — **ток** и **напряжение**. Необходимо объяснить эти понятия, не залезая глубоко в физику. Проще всего объяснить какое-то явление на примере другого явления, с которым человек регулярно сталкивается в обыденной жизни, и ощущает его собственными органами чувств. Для электрических явлений лучше всего подходит **гидравлическая модель**<sup>5</sup>. В **ГМ** электрические явления заменяются условными водопроводными: **проводник** — труба, **электрический ток** — поток воды в этой трубе, **резистор** — сужение сечения трубы, препятствующее течению тока, **диод** — клапан, открывающийся только в одну сторону, и т.п. В результате маловразумительные абстрактные понятия физики электрических явлений превращаются в почти ощутимые потоки и давление воды: все мы ежедневно пользуемся водопроводом, даже в глухих регионах найдется бочка или хотя бы дырявое ведро.

Таким образом, любому школьнику можно объяснить эти понятия:

---

<sup>5</sup> пример применения гидравлической модели, осторожно, г\*\*\*осайт

Электрическое **напряжение** — давление “электрической воды” в проводнике-трубе.  
Измеряется в [В]ольтах.

Как и обычное давление, **напряжение — разностная величина, так как измеряется между двумя точками** электрической цепи. Для обычного давления за опорную величину принимают атмосферное давление<sup>6</sup>. Для электричества напряжение измеряют тоже между двумя точками, или между точкой и **общим проводом**, принимаемым за 0. В физике напряжение определяют также как **разность потенциалов** между двумя точками.

Если взять проводник, и разделить его условной плоскостью, перпендикулярной проводнику, в области пересечения этой плоскости и проводника получается область — **сечение проводника**.

Сечение проводника легко увидеть и реально — достаточно взять толстый одножильный сплошной провод, и аккуратно отпилить (не откусить) его точно поперек. Поверхность такого отпила и будет сечением.

Электрический **ток** — количество “электрической воды”, проходящей через сечение проводника-трубы в единицу времени. **Сила тока** измеряется в [А]мперах.

$$j = \frac{I}{S} \quad (3.1)$$

где  $j$  **плотность тока**, А/м<sup>2</sup>  
 $I$  **сила тока**, Ампер  
 $S$  **площадь сечения проводника**, м<sup>2</sup>

<sup>6</sup> не ощущаемое человеком, так как оно уравновешено таким же внутренним давлением крови

Чем выше плотность тока, тем больше нагревается проводник (подробнее см ??), поэтому она учитывается при выборе провода, и ширины проводников печатных плат. В этом случае часто используется численно другая величина —  $A/\text{мм}^2$ .

## 3.5 Проводники и изоляторы, сопротивление и проводимость



1. <https://www.youtube.com/watch?v=q4I1hZ5YQ2w> Электрическое сопротивление проводника.
2. <https://www.youtube.com/watch?v=S1x0EEaQ4Cg> Удельное сопротивление  
<https://www.youtube.com/watch?v=FCkR-3YE5Ac>

Используя гидравлическую модель, точно так же просто можно объяснить понятие **проводимости**. Любой материал можно представить как кусок вещества, имеющий пористую структуру, например как поролон или песок. Через этот пористый материал течет “электрическая вода”, т.е. ток. Это течение вызвано действием напряжения, **приложенного** к концам проводника (источником питания). Напряжение проталкивает электричество через материал, поэтому чем больше напряжение, тем выше сила проходящего тока. Каждый материал имеет свою “пористость”, чем она больше, тем больше **проводимость**:

$$I = G U \quad (3.2)$$

где  $I$  сила тока, А  
 $G$  проводимость, См (сименс)  
 $U$  напряжение, В

Чаще вместо проводимости используют обратную величину — **сопротивление**:

$$R = \frac{1}{G} = G^{-1} \quad (3.3)$$

где  $G$  проводимость, См (сименс)  
 $R$  сопротивление, Ом

Соответственно, формула 3.2 превращается в описанный в любом школьном учебнике физики **закон Ома для постоянного тока**:

$$I = \frac{R}{U} \quad (3.4)$$

В зависимости от проводимости или сопротивления материалы делят на несколько видов:

	$G$ , проводимость	$R$ , сопротивление
сверхпроводник	$\infty$	0
проводник	большая	низкое
изолятор, диэлектрик	низкая	высокое

В зависимости от внешних условий: температуры, давления, агрегатного состояния вещества<sup>7</sup>, примесей — одно и то же вещество может быть как проводником, так и изолятором.

Типичные проводники: металлы, ионизированный газ (плазма), растворы солей в воде (электролиты), **влажное** дерево

Типичные диэлектрики: пластики, стекло, керамика, **сухое** дерево, вакуум, газы в т.ч. воздух.

В электронике очень часто нужно иметь в определенном месте схемы заданное сопротивление, для этого выпускаются специальные элементы с **точно калиброванным сопротивлением** между **выводами**: **резисторы**,

<sup>7</sup> плазма, газ, жидкость, твердое тело

иногда применяют устаревшее название **сопротивление** (часто проволочное). Их делают из керамики, на которую наматывают проволоку, или напыляют тонкую пленку из специальных металлических сплавов с высоким сопротивлением<sup>8</sup>. Иногда используется только кусочек такого сплава без керамики. Для качественной пайки выводы резисторов делают из хорошо проводящего металла, хорошо смачиваемого расплавленным припоем.

## 3.6 Диэлектрическая и тепловая прочность изоляции

Для диэлектриков существует специальный параметр, характеризующий их способность работать как изолятор — **диэлектрическая прочность**.

Чем выше диэлектрическая прочность, тем большее напряжение способен выдерживать изолятор без разрушения.

При повышении температуры диэлектрическая прочность уменьшается

Именно поэтому



Запрещается использовать любые удлинители в свернутом состоянии, или использовать электрический провод, свернутый бухтой (катушкой)

---

<sup>8</sup> часто в состав входят никель, хром, вольфрам



Не допускается нагрев любых частей электронной схемы или элементов конструкции выше  $40..50^{\circ}\text{C}$

Для любого диэлектрика существует некоторое значение напряжения, при котором он начинает пропускать ток. При прохождении тока материал нагревается (??). В результате нагрева проходящим током или от соседних соприкасающихся поверхностей у диэлектрика увеличивается проводимость, что приводит к еще большему проходящему через него току, и дополнительному нагреву.

Если тепло не отводится во внешнюю среду (например в середине катушки провода смотанного удлинителя), температура продолжает повышаться. В некоторый момент температура достигает значения, при котором слой изоляции размягчается, утрачивает механическую прочность, и соседние проводники сближаются или соприкасаются с плохим контактом (происходит **короткое замыкание**).

**В месте плохого контакта или большого тока температура резко повышается** до значений, когда диэлектрик воспламеняется или резко меняет химический состав, распадаясь **с выделением ядовитых соединений**<sup>9</sup> и полностью утрачивая изолирующие свойства как в электрическом, так и в механическом смысле.

Самый опасный вариант, и типичный случай **выгорания проводки** — **ток короткого замыкания** слишком маленький для срабатывания защитных устройств (**предохранитель, автомат, устройство защитного отключения, УЗО**), но достаточный для нагрева изоляции до температуры химического распада или воспламенения. Аварийного выключения не происходит, а проводка продолжает гореть до победного конца.

## 3.7 Тепловое действие тока. Мощность

---

<sup>9</sup> для типичной изоляции из ПВХ — сложные токсичные соединения с содержанием хлора



1. <https://www.youtube.com/watch?v=2LWAI0HRI8s> Нагревание проводников электрическим током  
Закон Джоуля Ленца
2. <https://www.youtube.com/watch?v=Mbt40TgBRuw> Мощность электрического тока

## 3.8 Масштабные множители

<https://www.youtube.com/watch?v=i3ABWmCl1EI> Перевод единиц измерения

Практически для всех единиц измерения необходимо использование **масштабных множителей**, когда величины численно или слишком большие (много цифр до запятой), или слишком маленькие (много цифр после запятой):

пико	p	$10^{-12}$
напо	n	$10^{-9}$
микро	u, $\mu$	$10^{-6}$
милли	m	$10^{-3}$
кило	k, K	$10^3$
мега	M	$10^6$
гига	G	$10^9$

$$10 \text{ mA} = 10 \times 10^{-3} \text{ A} = 10 \times 0.001 \text{ A} = 0.010 \text{ A}$$

$$5 \text{ КОм} = 5 \times 10^3 \text{ Ом} = 5 \times 1000 \text{ Ом} = 5000 \text{ Ом}$$

## 3.9 Использование мультиметра



<https://www.youtube.com/watch?v=BEgvm4o-u2Q>  
<https://www.youtube.com/watch?v=UMwYLwsPgCY>  
1. <https://www.youtube.com/watch?v=GBuGSj1uPGk>  
2. <https://www.youtube.com/watch?v=VJ3RBS42IVY>  
<https://www.youtube.com/watch?v=q3R4s6WE1cI>

Возьмите мультиметр (10.1) и измерьте напряжение на резисторе, близко ли оно к 7 В ? Также измерьте ток через диод, не превышает ли он допустимый ?

Напряжение измеряется в [В]ольтах подключением измерительного прибора (мультиметра) параллельно элементу, при этом нужно

- режим измерения включить на режим измерения постоянного напряжения  $V- / DCV$  (или переменного, обозначается как  $V \sim / ACV$ ) а
- диапазон измерения выставить на максимальное значение напряжения
- Последовательно уменьшая диапазон измерения, найдите диапазон, в котором мультиметр показывает наибольшее количество знаков после запятой.

Если диапазон слишком большой, прибор покажет значение в районе 0, а если слишком низкий — выведет [1]. Обычно работают в диапазоне, соответствующем максимальному напряжению питания (в нашем случае

20 В<sup>10</sup>), иногда для слабых напряжений переключаясь на одну..две ступени ниже. Но — возможны случаи<sup>11</sup> когда напряжение в части схемы на порядки (в десятки..тысячи раз) превосходит напряжение питания.

Ток измеряется в [А]мперах подключением измерительного прибора (мультиметра) последовательно с элементом в разрыв цепи, при этом нужно

- режим измерения включить на режим измерения постоянного тока  $A—/DCA$  (или переменного, обозначается как  $A \sim /ACA$ ) а
- диапазон измерения выставить на максимальное значение тока
- Последовательно уменьшая диапазон измерения, найдите диапазон, в котором мультиметр показывает наибольшее количество знаков после запятой.

Обратите внимание, что напряжение измеряется вольтметром на полностью собранной схеме, а для измерения тока нужно изменять схему, включая в нее амперметр. Если измерения тока нужно проводить на готовом устройстве, иногда ставят 2хконтактный джампер (как на материнских платах компьютеров): при измерении амперметр подключают к его контактам, а потом джампер замыкают специальной съемной проводящей перемычкой. Этот прием вы можете использовать в своих устройствах для регулярного измерения тока потребления, и расчета потребляемой мощности:

$$W_{\text{потребляемая}}[\text{Ватт}] = U_{\text{питания}}[\text{Вольт}] \times I_{\text{устройства/элемента}}[\text{Ампер}] \quad (3.5)$$

---

<sup>10</sup> [V]olt = [B]ольт

<sup>11</sup> в любых схемах содержащих индуктивности: катушки провода, трансформаторы, электродвигатели, динамики и т.п.

Переключив мультиметр в режим прозвонки (тестера), можно проверить

- наличие электрического соединения между двумя точками обязательно отключенной от источника питания схемы,
- исправность диода или светодиода,
- исправность конденсатора большой емкости и
- любые другие случаи когда нужно определить что две точки электрически соединены между собой, постоянно или временно.

Если между щупами мультиметра в режиме прозвонки<sup>12</sup> сопротивление протекающему току не превышает 1 КилоОм<sup>13</sup>, раздается звуковой сигнал.

Если щупы подключить к исправному разряженному конденсатору большой емкости (“электролиту”), раздается короткий целчок или даже “пик”.

Тестером также можно определить тип и цоколёвку<sup>14</sup> транзистора; как это сделать описано в ??.

Диоды проверяются двумя подключениями в разной полярности — при красном щупе на аноде (+) диода (в прямой полярности включения) диод проводит ток, а в обратной полярности нет (красный щуп на катоде (-)).

Проверьте светодиод: отключите его из схемы, и проверьте как диод. Если светодиод исправен, в прямой полярности светодиод будет очень слабо светиться.

---

<sup>12</sup> = тестера

<sup>13</sup> см. паспорт на прибор

<sup>14</sup> “раскопытку”

## 3.10 Определение сопротивления резистора по цветовому коду

Когда берете резистор, проверьте его **номинал** (значение). В наших схемах каждый резистор имеет свою цель, и значение выбирается в зависимости от того, хотим ли мы больший или меньший ток в этой части цепи. Чем выше **номинал** резистора, тем меньше ток. Чем ниже номинал резистора, тем выше ток. Для **выводных** резисторов, которые мы будем использовать на макетках, маркировка наносится на корпус в виде набора цветных полосок:

Цветовая маркировка на 5 цветных полосок:































цифра	цифра	цифра	множитель	точность
-------	-------	-------	-----------	----------

Цифра: 0 1 2 3 4 5 6 7 8 9

0	1	2	3	4	5	6	7	8	9
■	■	■	■	■	■	■	■	■	□

Множитель: ■  $10^0 = 1$  ■  $10^1 = 10$  ■  $10^2 = 100$  ■  $10^3 = 1000$   
■  $10^4 = 10000$  ■  $10^5 = 100000$  ■  $10^6 = 1000000$   
■  $10^{-1} = 0.1$  ■  $10^{-2} = 0.01$

Точность: ■  $\pm 1\%$  ■  $\pm 2\%$  ■  $\pm 5\%$  ■  $\pm 10\%$

-[						]-	$100 \times 10^4 \pm 1\%$	1M	1 миллион Ом	1M $\Omega$	1 000 000 Ом
	1	0	0	4	1						
-[						]-	$100 \times 10^2 \pm 2\%$	10k	10 тысяч ом	10,000 Ом	10k $\Omega$
	1	0	0	2	2						
-[						]-	$100 \times 10^1 \pm 5\%$	1k	1 тысяча ом	1000 Ом	1k $\Omega$
	1	0	0	1	5						
-[						]-	$390 \times 10^0 \pm 1\%$	390R	390 Ом	390 $\Omega$	
	3	9	0	0	1						
-[						]-	$100 \times 10^0 \pm 1\%$	100R	1000 Ом	100 $\Omega$	
	1	0	0	0	1						
-[						]-	$470 \times 10^{-1} \pm 1\%$	47R	47 Ом	47 $\Omega$	
	4	7	0	-1	1						

## Часть III

САПР электронных устройств KiCAD

# Глава 4

## eeschema: редактор электрических схем

обеспечивает:

- создание однолистовых и иерархических схем,
- проверку их корректности ERC (контроль электрических правил),
- создание списка электрических цепей netlist для редактора топологии платы pcbnew или для spice-моделирования схемы,
- доступ к документации на используемые в схеме электронные компоненты (datasheet).



## Часть IV

Расчет схем и моделирование в ngSPICE

на основе статьи

1

2

## Electronic circuit simulation with gEDA and NG-Spice by Example

© Andreas Fester

**SPICE:** [S]imulation [P]rogram with [I]ntegrated [C]ircuit [E]mphasis — пакет программ симуляции и расчета электронных схем, была создана для моделирования **интегральных микросхем**, расчета режимов работы, оптимизации, и предсказания поведения.

SPICE может выполнять несколько видов схемотехнических расчетов, самые важные из которых:

- Нелинейный анализ по постоянному току: вычисление передаточной характеристики по постоянному току
- Нелинейный анализ переходных процессов: вычисление токов и напряжений как функции времени в условиях большого сигнала
- Линейный АС анализ: вычисление выхода как функции от частоты. Выводится **bode plot**
- Анализ шума
- Расчет чувствительности
- Анализ искажений
- Фурье-анализ: вычисление и отображение частотных спектров
- Анализ Монте-Карло

---

<sup>1</sup> копия: [http://www.mithatkonar.com/wiki/doku.php/kicad/kicad\\_spice\\_quick\\_guide](http://www.mithatkonar.com/wiki/doku.php/kicad/kicad_spice_quick_guide)

<sup>2</sup> копия: <http://physics.gmu.edu/~rubinp/courses/407/ngspice.pdf>

# Глава 5

## Доступные SPICE-пакеты

Первоначально SPICE был разработан в Университете Беркли. Другие версии SPICE являются форками этой реализации, и сейчас существует несколько вариантов:

- **ngSPICE**: самый доступный бесплатный OpenSource SPICE-движок.
- <http://www.ngspice.com/> — on-line вариант **ngspice**, удобен для начального обучения
- **LT-SPICE**: Популярная бесплатная коммерческая версия от Linear Technology для Windows. Работает в **WINE**. Поставляется в комплекте с графической оболочкой, но требуется проверка файлов расчетных заданий, которые она создает.
- **gnucap**<sup>1</sup>: Не совсем SPICE, но пытается быть синтаксически совместимой.
- **SpiceOpus**: Коммерческая, но удобная, особенно в плане вывода графиков.

---

<sup>1</sup> уже включен в винدوزную сборку KiCAD

- PSpice: ☐Windows-only, дорогой коммерческий пакет, стандарт de-facto для профессионального применения в USA в составе тяжелых EDA-продуктов. Они традиционно предоставляют обрезанную [gratis](#)-версию для студентов. Также имеет в комплекте GUI.
- MultiSim, OrCAD, DesignLab,... коммерческие EDA-пакеты, содержит в составе одну из версий **PSPICE**
- Какие-то еще?

В этом разделе описан **ngSPICE**, который был написан с целью полностью переписать оригинальную реализацию Беркли SPICE. Сейчас он все еще содержит часть кода Беркли, но в этом коде было исправлено множество ошибок. Важно понять, что каждая реализация SPICE может вести себя особенно в некоторых случаях: некоторые из них более совместимы между собой, другие менее. Всегда важно прочитать документацию, которая поставляется с конкретной версией SPICE. Дистрибутив **ngSPICE** поставляется с детальным руководством пользователя, а этот раздел поможет вам начать. Хорошо иметь под рукой полное руководство при чтении этого раздела, так как интересующие вас команды там рассмотрены детальнее.

## 5.1 Установка **ngSPICE** под ☐Windows

```
☐ + R >> http://ngspice.sourceforge.net/download.html >> ng-spice-rework >> ngspice-xx_xxxxxx.zip
unzip ngspice-xx_xxxxxx.zip >> C:/spice
☐ >> Компьютер >> Свойства >> Дополнительные параметры системы
Переменные среды >> PATH=...;C:/spice/bin
```

## 5.2 Установка **LT-SPICE** (только ☐Windows)

```
☐ + R >> http://www.linear.com/designtools/software/
```

## 5.3 Установка ngSPICE под Linux

```
root# aptitude install ngspice
```

Пакет **ngSPICE** также может быть легко собран и установлен компиляцией их исходников. После загрузки архива, соберите пакет для вашего дистрибутива:

```
tar xvfz ng-spice-rework-15.tgz
cd ng-spice-rework-15
./configure --with-readline=yes
make
checkinstall
```

Убедитесь что в системе присутствует библиотека **GNU readline**, и она включена в опциях **configure**: ее использование делает интерфейс командной строки более комфортным. Подробнее сборка **ngSPICE** описана в его руководстве в составе дистрибутива.

Сборка **ngSPICE** для azLinux описана в разделе ??.

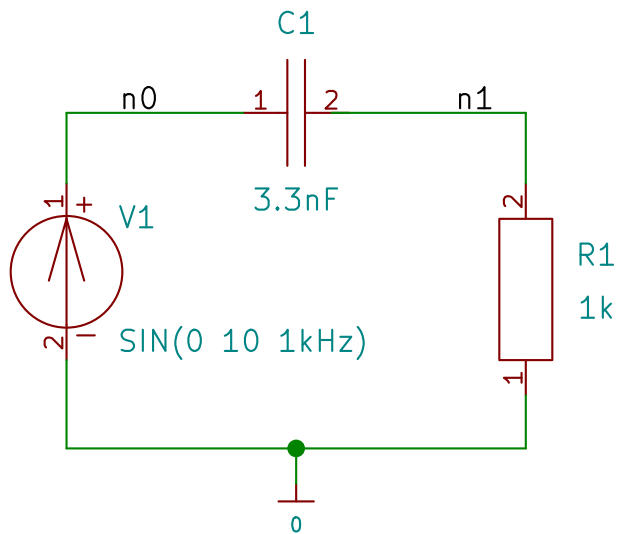
# Глава 6

## Пошаговый пример использования

Рассмотрим очень простой пример симуляции электронной схемы. В основном вы будете рисовать ваши схемы в **eeschema**, но интерактивная симуляция пока не реализована.

### 6.1 Рисуем схему в **KiCAD**

Схема, которую мы хотим рассчитать — простой **RC-фильтр**. Эта схема была выбрана, так как она содержит очень ограниченное количество элементов, и поэтому понятна: источник напряжения, резистор и конденсатор. Использование редактора схем вам уже должно быть знакомо из предыдущего раздела (4). Запустите **eeschema** и нарисуйте схему:



Простой RC-фильтр

Имена, указанные на проводниках — **имена цепей**. Они могут не указываться, если вам не нужно на них ссылаться в расчете. При экспорте списка цепей из **eescema** им будут присвоены уникальные имена. Цепь может иметь любое имя, за некоторым исключением: **в списке должна быть одна цепь с именем [0], она подключается к общему проводу (земле)**. Рисуя схему, поместите на нее один или несколько элементов **[0]** из библиотеки **spice**.

**V1** — **независимый источник напряжения**. Его значение задано в виде выражения SIN(0 10 1kHz), создающего синусоидальный (SIN) сигнал со смещением (0) вольт, амплитудой (10) вольт и частотой (1kHz).

Рисуйте вашу схему, соблюдая несколько рекомендаций:

- Для именованных цепей используйте **глобальные метки** вместо локальных. В списке цепей глобальные идентификаторы цепей включаются как есть, а локальные метки модифицируются, что делает сложным последующие ссылки на них при SPICE-моделировании.

- Используйте компонент [0] из библиотеки **spice**, вместо обычного компонента [GND]: "0" официальное имя главной земли в файлах PSPICE. Некоторые SPICE-движки умеют транслировать  $GND \rightarrow 0$ , другие нет.

## 6.2 Создание списка цепей

Входными данными для симуляции является **список цепей (netlist)**. Список цепей создается через экспорт из **eeschema**. Если вы используете программу рисования схем, посмотрите в ее документации, умеет ли она экспорт в SPICE(.cir), и как это сделать.

- Кликните кнопку или пункт меню **Сформировать список цепей**.
- Выберите вкладку **Spice**, и убедитесь что включен крыжик ☒ **Формат по умолчанию**. Вам нужно сделать это только один раз, настройки запоминаются.
- Нажмите кнопку **Сформировать**

Если вы хотите запускать **ngSPICE** из диалога экспорта:

- Заполните полный путь с программе симуляции, типа **C:/spice/bin/ngspice.exe** со всеми путями и расширениями, **KiCAD** пока не научился запускать симулятор через PATH.
- Нажмите кнопку **Запустить симулятор**.

В результате экспорта будет создан файл



```
1 * RC_filter
2
3 *Sheet Name:/
4 R1 0 /n1 1k
5 C1 /n0 /n1 3.3nF
6 V1 /n0 0 SIN(0 10 1kHz)
7
8 .end
```

Формат нетлиста SPICE прост: каждая строка содержит один элемент схемы. Первый столбец каждой строки содержит имя элемента, затем идут имена цепей, которым подключен каждый вывод, последним идет значение элемента. Строки, начинающиеся с `[*]` — комментарии. В нашем примере строка, начинающаяся с **V1**, описывает источник напряжения, подключенный к цепям **n0** (вывод 1) и **0** (вывод 2), значение `SIN(0 10 1kHz)`. Точно также заданы конденсатор и резистор. Так как цепи **n0,n1** заданы именами, перед ними стоит `[/]`.

Первая и последняя строки имеют для **ngSPICE** особое значение: при чтении нетлиста **ngSPICE** считает первую строку названием схемы. Последняя строка должна содержать токен **.end**.

Так как формат файла нетлиста настолько прост, его можно легко создать вручную из любого текстового редактора. Некоторые статьи о SPICE-симуляции даже начинаются с такого способа, но он действительно неудобен для работы. Используя **eeschema** или другой редактор схем, намного проще изменять схему, и она может быть легко распечатана или включена как иллюстрация в документацию. Единственный реальный вариант, когда нужно работать напрямую в файлом — если вам вдруг понадобится сформировать его автоматически, например при анализе паразитных емкостей и индуктивностей печатной платы, которые определяются формой печатных проводников.

## 6.3 Запуск симуляции

Теперь вы готовы симулировать схему. Прежде всего нам нужно решить, какие виды расчетов, которые умеет делать SPICE, нас интересуют:

- **Анализ переходных процессов** показывает поведение схемы во времени.
- **Расчет по переменному току (AC)** дает изменения работы схемы с изменением (входной) частоты.
- **Параметрическая симуляция** позволяет анализировать изменения в работе схемы при изменении одного или нескольких параметров, например изменении частоты источника и емкости конденсатора.

Для начала посмотрим как ведет себя входное напряжение во времени. Мы хотим выполнить анализ переходных процессов в схеме, и вывести напряжение между сетями **n0** и **0**.

Запускаем **ngSPICE**:

```
$ ngspice
```

```
spinit found in c:\spice\share\ngspice\scripts\spinit
```

```
*****
```

```
** ngspice-24 : Circuit level simulation program
```

```
** The U. C. Berkeley CAD Group
```

```
** Copyright 1985-1994, Regents of the University of California.
```

```
** Please get your ngspice manual from http://ngspice.sourceforge.net/docs.html
```

```
** Please file your bug-reports at http://ngspice.sourceforge.net/bugrep.html
```

```
** Creation Date: Jan 30 2012 22:58:51
```

```
*****
```

```
ngspice 1 ->
```

Теперь нам нужно загрузить нетлист (выводится заголовок: первая строка файла):

```
ngspice 1 -> source RCfilter.cir
```

```
Circuit: * eeschema netlist version 1.1 (spice format) creation date: 26.12.2014 16:15:26
```

Так как мы задали для входного напряжения частоту 1 КГц, период  $T = 1/F = 0.001 \text{ с} = 1 \text{ мс}$ . Мы хотим увидеть как входное напряжение меняется за первые 5 периодов, т.е. 5 мс. Запускаем симуляцию следующей командой:

```
ngspice 2 -> tran 0.01ms 5ms
```

```
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
```

```
Warning: v1: no DC value, transient time 0 value used
```

```
Initial Transient Solution
```

```
-----
```

Node	Voltage
----	-----
/n1	0
/n0	0
v1#branch	0

```
No. of Data Rows : 512
```

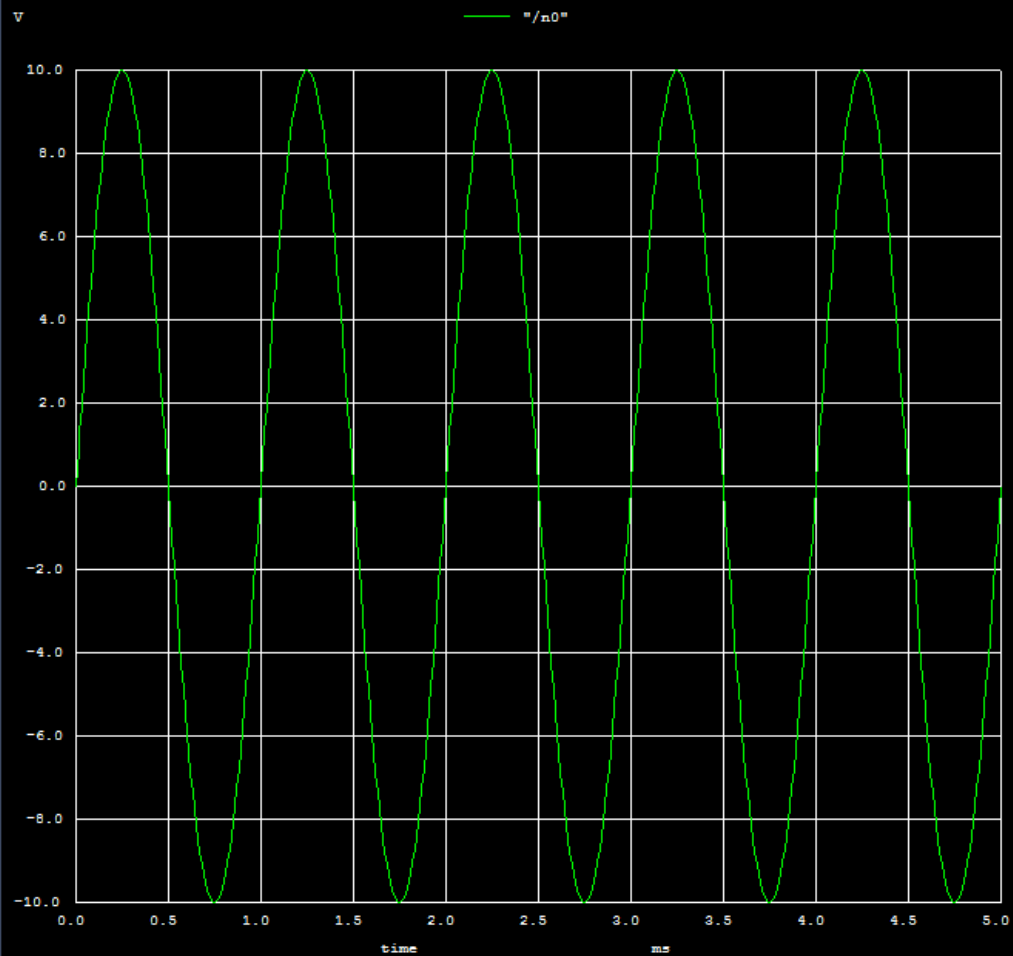
Первый параметр tran определяет **шаг расчета**, второй — **конечное значение времени**. Если не указан третий параметр, начальное время равно 0, иначе третий параметр указывает ненулевое **начальное время**. Ну, это все ☺. Симуляция выполнена. Теперь нам нужно увидеть результат симуляции.

## 6.4 Просмотр результата расчета

SPICE создал таблицы с рассчитанными значениями: 512 значений для каждого узла схемы. Для простого просмотра чисел выполним команду (не забудьте про кавычки, без них не работает если первый символ [/]):

```
ngspice 3 -> plot "/n0"
```

tran1: \* eeschema netlist version 1.1 (spice format) creation date: 26.12.2014 16:15:26



Эта команда вывела напряжение при переходном процессе на цепи [n0].

Как вы заметили, диаграмма сигнала отображается на черном фоне. Если вам нужны другие цвета, например для вставки в документацию, их можно переопределить:

```
ngspice 12 -> set color0 = white
ngspice 13 -> set color1 = black
ngspice 14 -> set color2 = green
ngspice 3 -> plot "/n0"
```

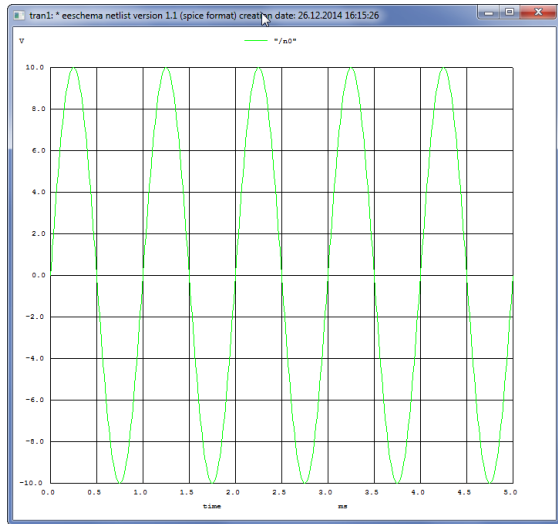
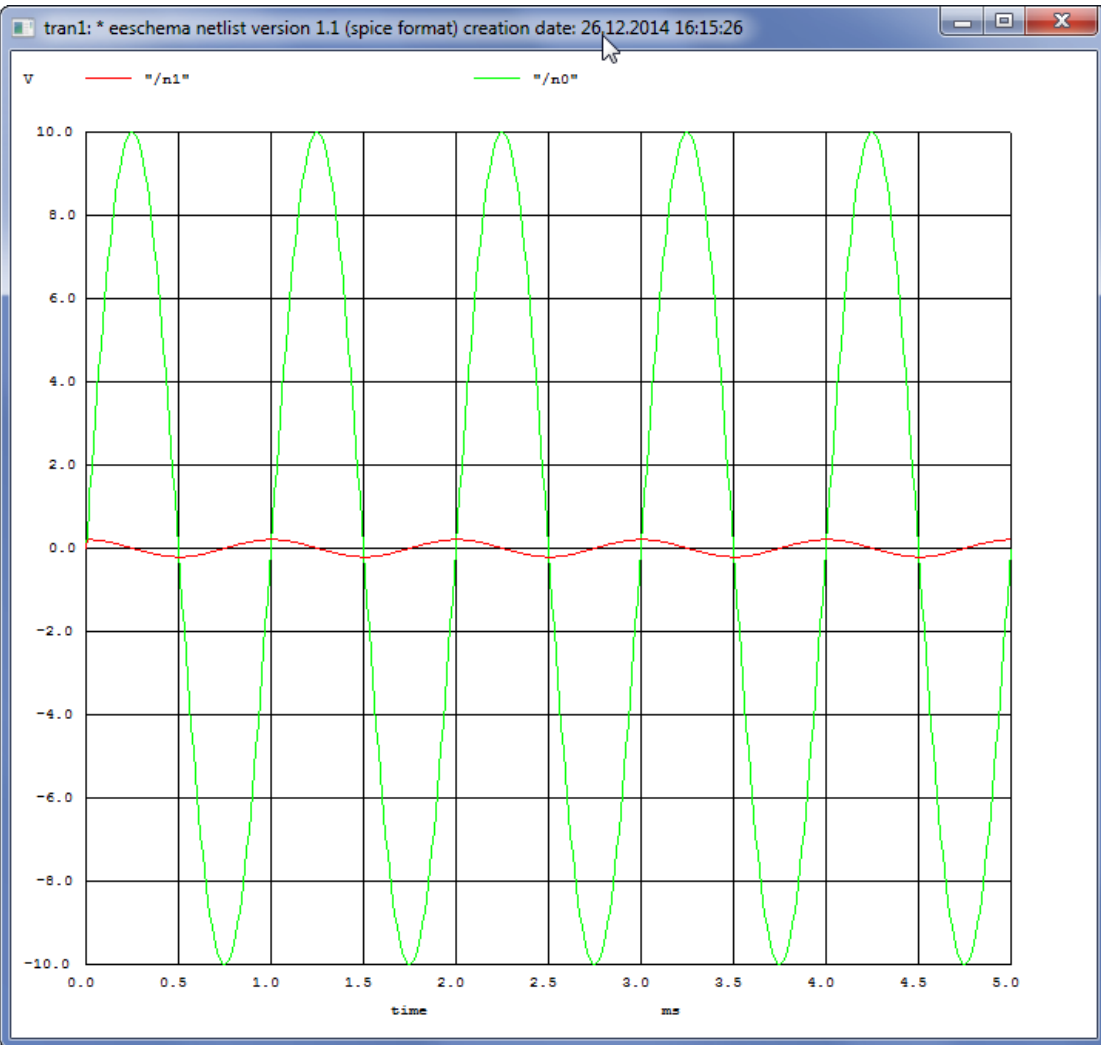


Диаграмма показывает форму входного сигнала, как мы и ожидали, но она нас мало интересует, так как мы ее и задали. Нам интереснее например [напряжение на резисторе](#), кроме того мы попробуем [сравнить два сигнала](#). Это легко сделать указав два имени цепи:

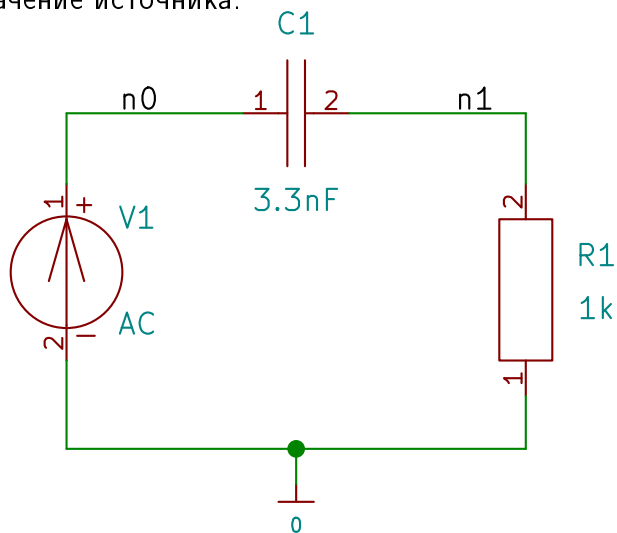
```
ngspice 31 -> plot "/n0" "/n1"
```



## 6.5 Расчет АЧХ по переменному току (АС симуляция)

Сигнала на резисторе почти не видно. Теперь вопрос: какие частоты попускает наш фильтр? Для определения этого теперь выполним **расчет по переменному току (АС симуляцию)**. Команда для этого `ac ( DEC j OCT j LIN ) N FStart FEnd`.

FStart и FEnd — соответственно начальная и конечная частота. Необязательные параметры DEC, OCT или LIN указывают способ изменения частоты: декадно, октавно или линейно. Если выбрана октавная или декадная **вариация частоты**, то параметр N задает число частот на декаду или октаву. Для выполнения **АС анализа** должен быть изменен источник сигнала: сейчас он определен как синус с амплитудой 10 В и частотой 1 КГц. Для анализа это должен быть **источник переменного напряжения**. Снова запускаем **eeschema** и меняем значение источника:



Создаем нетлист, загружаем его и запускаем команду АС анализа:

```
$ ngspice ACanaliz.cir
```



```
ngspice 1 -> ac lin 1000 0.1 250kHz
```

```
Doing analysis at TEMP = 27.000000 and TNOM = 27.000000
```

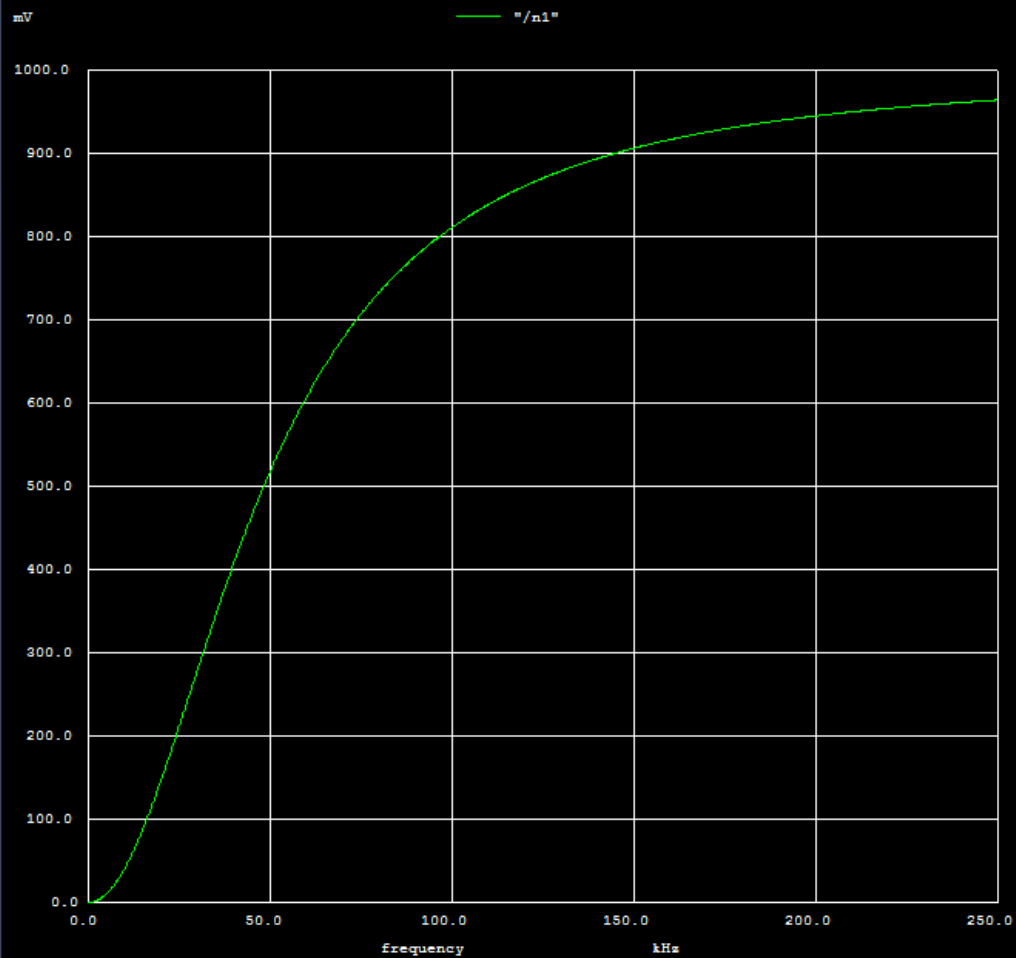
```
Warning: v1: has no value, DC 0 assumed
```

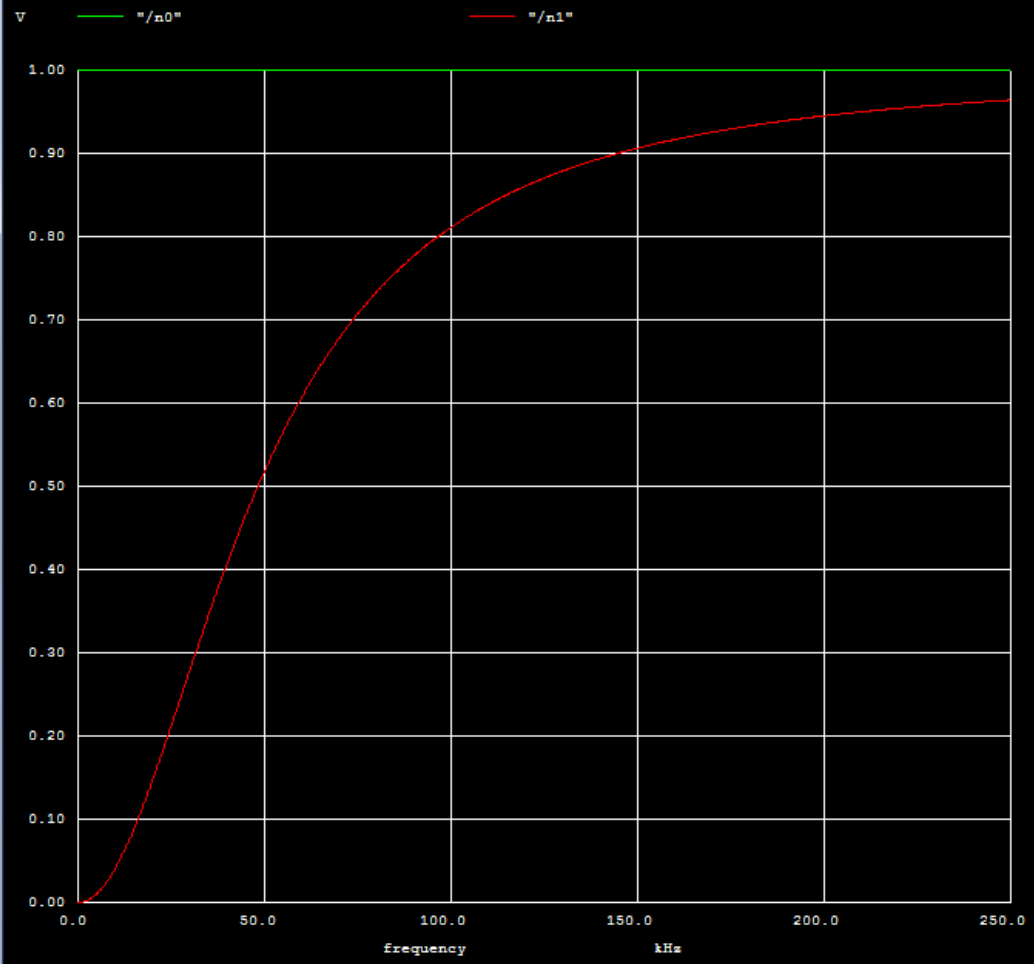
```
No. of Data Rows : 1000
```

```
ngspice 2 -> plot "/n1"
```

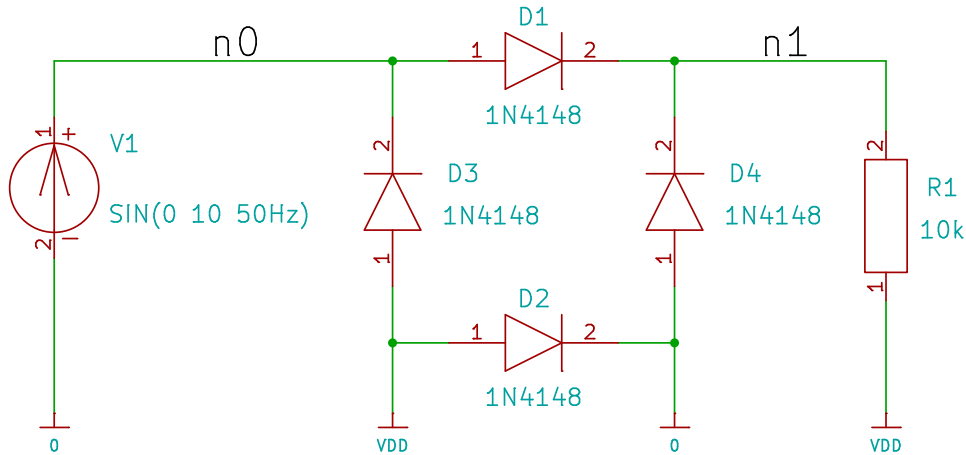
```
ngspice 3 -> plot "/n0" "/n1"
```

Эта команда выполняет **линейный АС анализ** от (почти) 0 Гц до 250 КГц. Результат можно увидеть как напряжение для источника, так и напряжение на **R1**:





## 6.6 Симуляция полноволнового выпрямителя



–PSPICE  
\* text before netlist

+PSPICE  
\* text after netlist  
.control  
tran 0.01ms 50ms  
set hcopydevtype=postscript  
set hcopypscolor=true  
set color0=white  
set color1=black  
set color2=rgb:F/0/0  
set color3=rgb:0/F/0  
set color4=rgb:0/0/F  
hardcopy RectifierPlot.eps "/n0" "/n1"  
quit  
.endc

Rectifier.cir

```
1 * Rectifier
2
3 * text before netlist
4
5 *Sheet Name:/
6 D4  0 /n1 1N4148
7 D2  0 0 1N4148
8 D3  0 /n0 1N4148
9 D1  /n0 /n1 1N4148
```

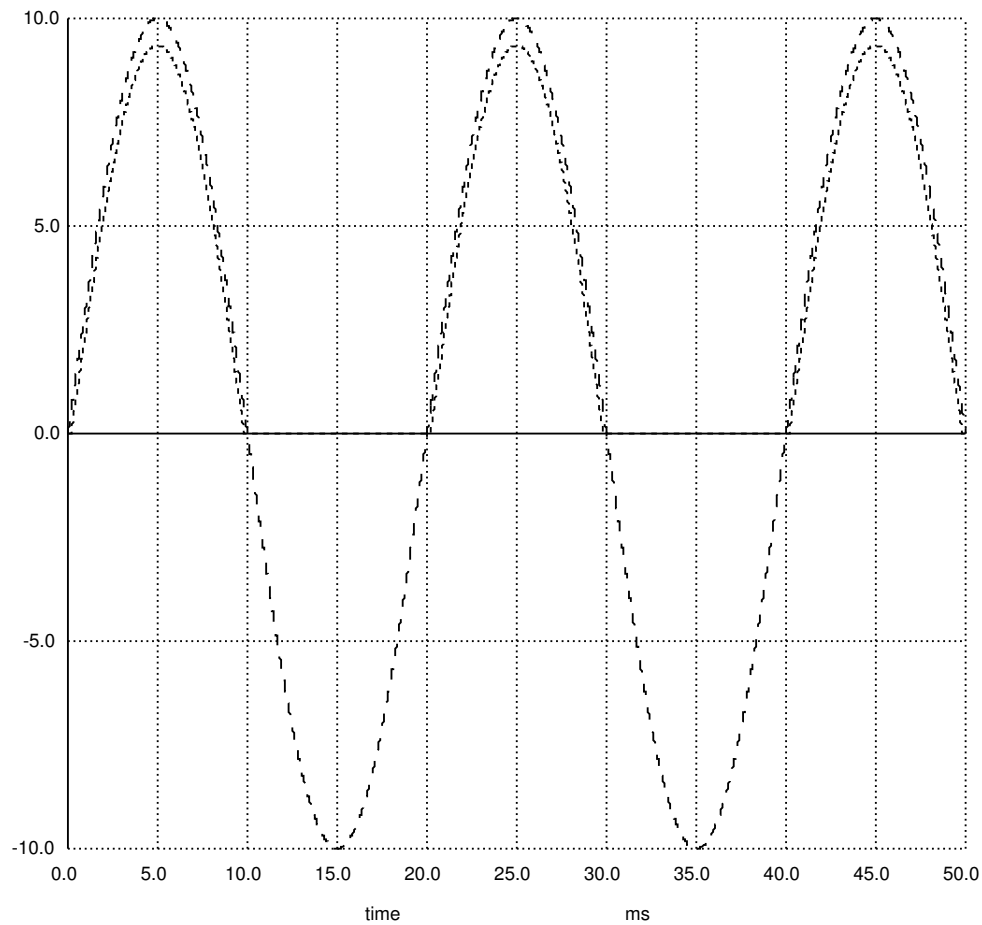
```
10 R1 0 /n1 10k
11 V1 /n0 0 SIN(0 10 50Hz)
12
13 * text after netlist
14 .control
15 tran 0.01ms 50ms
16 set hcopydevtype=postsript
17 set hcopypscolor=true
18 set color0=white
19 set color1=black
20 set color2=rgb:F/0/0
21 set color3=rgb:0/F/0
22 set color4=rgb:0/0/F
23 hardcopy RectifierPlot.eps "/n0" "/n1"
24 quit
25 .endc
26
27 .end
```

Подробнее использованные здесь приемы рисования схемы описаны в 7.

Кратко: была применена возможность вставки в нетлист текстовых блоков до и после списка элементов. При запуске **ngspice** из **KiCAD** будет автоматически выполнен блок `.control/.endc`. Также для вывода графиков была использована команда `hardcopy`, предварительно настроенная на вывод в формате (Encapsulated) PostScript. Текущая версия **ngspice** не умеет цветной ввод, он должен быть починен в следующих версиях.

V      - - - - "n1"

- - - - "n0"



# Глава 7

## Настройка KiCAD для SPICE-моделирования

### 7.1 Библиотеки компонентов со SPICE-элементами

- Библиотека базовых SPICE-компонентов поставляется с KiCAD. Эта библиотека — хороший вариант для начальных экспериментов. Библиотека не подключена по умолчанию, вы должны сделать это вручную сами через менеджер библиотек. На Debian Linux это файл `/usr/share/kicad/library/pspice.lib`<sup>1</sup>
- Mithat Konar <webs@mithatkonar.com> разрабатывает (очень медленно) собственную библиотеку с некоторыми модификациями.
- В комплекте с этой книгой поставляются библиотеки, адаптированные для SPICE.

---

<sup>1</sup> PSpice — популярная коммерческая версия SPICE

## 7.2 Настройка проекта

1. Создайте новый проект как обычно.
2. Откройте **Eeschema** и удалите все библиотеки, подключаемые по умолчанию.
3. Вручную добавьте одну из SPICE-библиотек, или набор библиотек для этой книги. Обратите внимание, что SPICE-библиотека из поставки **KiCAD** по умолчанию не подключается к проекту.
4. Укажите расчетный SPICE-движок, который вы хотите использовать:

**eeschema** » Меню » Инструменты » Сформировать список цепей » Spice

☒ Формат по умолчанию

☐ Префикс обозначений

☒ Использовать имена цепей

вкладка Spice » Команда симулятора: » **xterm -e ngspice**

Список цепей

## 7.3 Как это работает

1. Укажите режимы симуляции, которые вы хотите выполнить, и генерацию вывода, который хотите отобразить, добавив на схему текстовый блок (т.е. “комментарий”) с необходимыми директивами в синтаксисе SPICE и Nutmeg с некоторыми добавками. Например, для выполнения **расчета по постоянному току** и вывода сигнала в точке `out`, добавьте блок:

```
1 +PSPICE
2 .control
```



```
3 ac dec 66 1kHz 120kHz
4 plot vdb(vout)
5 set units = degrees
6 plot vp(vout)
7 .endc
```

- Первая строка "+PSPICE " указывает kicadu добавить текст [в конец](#) сформированного .cir-файла. [В текущей версии KiCAD есть баг, который требует обязательного пробела после +SPICE.](#)
- Соответственно строка "-PSPICE " добавляет текст [в начало](#) .cir-файла.
- Для поборников OpenSource, не желающих видеть ссылка на коммерческий PSpice, предусмотрены директивы-синонимы  $\pm$ "GNUCAP ". Я думаю это то же самое что и  $\pm$ "PSPICE ", но не уверен на 100%, проверьте в документации.
- Да, вам потребуется немного изучить синтакис SPICE and Nutmeg. Это нетрудно.

2. Запустите симуляцию:



## Часть V

Разработка конструкции в САПР FreeCAD



<sup>2</sup> В среде специалистов ряда отраслей известна проблема создания полноценной САПР в рамках OpenSource, и хотя FreeCAD ещё не является кандидатом на такую «полноценность», этот продукт может рассматриваться как одна из попыток создания базы для решения этой проблемы. Разработчик FreeCAD Юрген Ригель, работающий в корпорации DaimlerChrysler, позиционирует свою программу как первый бесплатный инструмент проектирования механики (сравнивая свой продукт с такими развитыми проприетарными системами как CATIA версий 4 и 5, SolidWorks), созданный на основе библиотеки **Open CASCADE**. Цель программы — предоставить базовый инструментарий этой библиотеки в интерактивном режиме.

Следует отметить, что имеет место ещё один программный продукт имеющий название freeCAD, его разработчик — Aik-Siong Koh, и он не связан с FreeCAD'ом Юргена Ригеля.

---

<sup>2</sup> копияста: [https://ru.wikipedia.org/wiki/FreeCAD\\_\(Juergen\\_Riegel%27s\)](https://ru.wikipedia.org/wiki/FreeCAD_(Juergen_Riegel%27s))

<sup>3</sup> FreeCAD — CAD/CAE приложение трёхмерного параметрического моделирования. Оно в основном сделано для механического проектирования, но также может быть использовано для любых других случаев, в которых вам нужно точно моделировать трёхмерные объекты с контролем над историей моделирования.

FreeCAD все еще находится в ранней стадии разработки, так что, хотя он уже предлагает Вам большой (и растущий) список функций, многого еще не хватает, особенно если сравнивать его с коммерческими решениями, и вы можете не найти его достаточно развитым для использования в производственной среде. Тем не менее, есть быстрорастущее сообщество пользователей-энтузиастов, и вы уже можете найти много примеров качественных проектов, разработанных с FreeCAD.

Как и все проекты с открытым исходным кодом, проект FreeCAD не единственный способ работы обеспеченный Вам его разработчиками. Это во многом зависит от роста его сообществу пользователей и разработчиков, доработки функций и стабилизации кода (да здравствует исправление ошибок!). Так что не забывайте об этом, когда начинаете использовать FreeCAD, если вам он нравится, вы можете непосредственно влиять и помочь проекту!

---

<sup>3</sup> копинаста: [http://www.freecadweb.org/wiki/index.php?title=Getting\\_started](http://www.freecadweb.org/wiki/index.php?title=Getting_started)

# Часть VI

Установка под  Windows

+ R > http://www.freecadweb.org/ > Download > Windows > **FreeCAD 0.14** > ...\_setup.exe

**FreeCAD 0.14.3700\_x86\_setup** > FreeCAD 0.14 License > I agree

Distination Folder > **C:/FreeCAD** > Install > Completed > Close

Программы > FreeCAD 0.14 > FreeCAD > > Закрепить в панели задач



[FreeCAD © Juergen Riegel, Werner Mayer, Yorik van Havre 2001-2011](#)

Версия	<b>0.14</b>
Редакция	<b>3700 (Git)</b>
Дата выпуска	<b>2014/07/13 11:34:36</b>
Операционная система	<b>Windows 7</b>
Word size	<b>32-bit</b>
Branch	<b>releases/FreeCAD-0-14</b>
Hash	<b>32f5aae0a64333ec8d5d160dbc46e690510c8fe1</b>

Лицензия ...

Скопировать в буфер обмена

OK

## Часть VII

Инструменты и электронное оборудование

## Глава 8

# Радиомонтажный инструмент

Пара надфилей, заточной камень на дрель, комплект сверел и несколько листов наждачки.

### 8.1 Pro'sKit

Отдельного обзора заслуживает инструмент и наборы Pro'sKit





PK-5308BM универсальный набор инструментов



1PK-616B Набор инструментов для электроники профессиональный



1PK-813B Набор базовых инструментов для электроники

По личному опыту: в 1РК-813В не хватает

- мелкого мультиметра,
- стриппера 1РК-3001Е,
- микрокусачек типа 8РК-30D,
- канифоли,
- ножа,
- настроечную отвертку заменить индикаторной.

## 8.2 Инструмент до 1000 В

Для электромонтажных работ обязательно приобретите комплект высоковольтного инструмента до 1000 В:



PM-911 Пассатижи 1 кВ



PM-917 Кусачки (бокорезы) 1 кВ

## 8.3 Хранение



**103-132D** Кассетница для деталей и компонентов



**SB-3428SB** Портативная кассетница для саморезов и т.п.

## 8.4 Радиомонтаж



8PK-30D Кусачки миниатюрные



1PK-709 Длинногубцы-кусачки



1PK-055S Длинногубцы изогнутые



1PK-29 Круглогубцы





1PK-101T Пинцет прямой



1PK-3001E Клеши для зачистки проводов  
прецизионные (стриппер)



PD-374 Тиски на струбцине

## 8.5 Прочие

Попалась интересная недорогая отвертка: аиксация четкая, исполнение очень неплохое, позволяет добраться до узких мест. Из минусов: ручка похоже не цельнометаллическая, при изломе есть риск распороть руку.



## Глава 9

# Паяльное оборудование

### 9.1 Паяльник

Паяльник — обязателен дешевый сетевой мощностью не менее 20 Вт, типа ЭПСН-25/220. Ограничитель мощности или регулятор температуры легко собрать самостоятельно.

Для сборки электроники хорошо также иметь маленький монтажный 12 В 8 Вт от паяльной станции ZD-927 (~100 р), без самой станции. Если не жалко 500 р, берите станцию ZD-927 целиком, внутри простейший регулятор мощности, и вам не понадобится источник питания на 12 В, который вы еще не сделали.



Паяльник ЭПСН-25/220



Паяльник 220В 25Вт, СВЕТОЗАР, SV-55310-25 230 р.



Паяльник 220В 25Вт ZD-721N 175 р.



Паяльник для станции ZD-927 12 В 8 Вт 85 р.

## 9.2 Паяльная станция

Из всего разнообразия для хоббита оптимальным являются паяльные станции Lukey 702/853D (3000+ р). Для работы или регулярного хобби паяльная станция с феном, а может даже и встроенным источником питания, вещь незаменимая, и не такая уж дорогая.



Паяльная станция ZD-927 520 р.



Паяльная станция LUKEY 702 3100 р.



Паяльная станция LUKEY 853D с источником питания 5200 р.



# Глава 10

## Измерительное оборудование

### 10.1 Мультиметр

Мультиметр — обязателен, без него работать невозможно<sup>1</sup>. Для совсем начинающего больше всего подойдет M32010.1.3 с автодиапазоном, когда освоитесь возьмете вторым прибором что-то из крупных серий M89х/MY6х с измерением температуры<sup>2</sup> или “рыльцемер” 10.5 (RLC).

---

<sup>1</sup> или собирать замену на паре измерительных головок тока/напряжения, и делителях

<sup>2</sup> иногда нужно для измерения температуры корпусов элементов, радиаторов, растворов если возитесь с электрохимией

### 10.1.1 Mastech M838



Простой, компактный, дешевый, с измерением температуры

## 10.1.2 Mastech M300



Простой, **очень компактный**, дешевый, в чехле очень удачно помещается в набор инструментов.

### 10.1.3 Mastech M320



То же что и M30010.1.2, но с [автодиапазоном](#), т.е. не требует переключения диапазонов измерения вручную. На любителя, возможно [удобен для совсем начинающих](#), но слишком медленен если требуется измерение меняющегося тока/напряжения.

10.2 Осциллограф

10.3 Логический анализатор

10.4 Генератор сигналов

10.5 Рыльцеметр RLC

# Глава 11

## Электроинструмент

## 11.1 Дрель



Дрель ударная сетевая  
Praktyl-R PID13D01 400 Вт (!)395 р.



Дрель безударная сетевая  
Интерскол Д-11/530ЭР (с БЗП) 1120 р.

Дрель — одноразовая китайчатина от 400 р. Продаются уже брендированные на Леруа Мерлен, наклейка «PID13D01 Ударная дрель 400 Вт, 13 мм». Скорость регулируется глубиной нажатия курка, крутилка на курке ограничивает глубину механически, фиксатор держит скорость близко к минимальной, запаха горелой пластмассы через несколько минут работы на холостом ходу нет.

По надежности рекомендуется Интерскол 1100+ р. Надежность Интерскола — не «китай», классика ДУ-580ЭР работает в хвост и гриву, используется криворукими студентами, лежит в подвале в пыли от точила, и никаких вопросов даже со щетками.

Если не планируете много сверлить бетон, **берите дрель без ударного механизма**: отсутствуют лишние продольные перемещения, что может быть важно при использовании в качестве шпинделя сверлильного станка, и механизации других технологических поделок.

У шуруповерта нет 43 мм шейки для фиксации, поэтому как средство электропривода он практически бесполезен, и нужен собственно для заворачивания большого количества саморезов. Хотя наличие ограничителя крутящего момента и малые габариты удобны при сверлении и сборке поделок.

Имея некоторое количество поделочного материала, кривые руки и особенно доступ к станочному оборудованию, можно сколхозить некоторое подобие настольных станочков 11.1 для механизации некоторых работ, используя дрель в качестве привода.

Главным элементом такой оснастки — зажим на шейку дрели 43 мм. Особых требований по его точности и качеству нет, т.к. сама шейка обычно пластиковая, и никакой доводки по круглости и параллельности оси инструмента не проходит.





## 11.2 Лобзик



Praktyl 350 Вт 356 р.



Makita 4329 2260 р.

Лобзик полезен при разделке стеклотекстолита, и изготовлении технологической мебели (стеллажи, рабочие столы и т.п.).

## 11.3 Жвигатель

Если у вас возникло желание механизировать изготовление механических деталей, а свободного доступа к настоящему станочному оборудованию нет, есть смысл рассмотреть изготовление самодельной механизированной оснастки типа 11.1, или даже самодельных станочков. В этом случае надо рассмотреть применения универсального привода.

Первый кандидат на место универсального электропривода достается той самой дрели, не забываем об обязательном наличии 43 мм монтажной шейки. Достоинство дрели как привода — прямое подключение к сети, встроенный редуктор, есть модели с простой регулировкой оборотов, есть резьба и отверстие под винт на валу, в комплекте есть патрон для зажима мелких деталей в точилке<sup>1</sup>.

Ограниченно доставаемые двигатели от стиральных машин, отличаются мощностью и оборотистостью, особенно от старых моделей. Часто доступны сразу с готовым шкивом на валу, который иногда проще использовать, чем снять.

Автозапчасти: привод печки Камаза, двигатель постоянного тока 24 В 50 Вт

Новые асинхронные двигатели АИРЕ 56 В2/В4 (3000/1500 об.) с заводским конденсатором, подключается к сети ~220 В, цена от 2500 р. С ростом размеров и мощности цена резко повышается. Следует обратить внимание на возможность монтажа на дополнительный фланцевый подшипниковый щит, (?) с моделями АИРЕ 80.

Для самодельных серилок и микроинструмента хороши китайские воздушные шпиндели постоянного тока с цанговыми патронами ER11. Требуют источник питания постоянного тока 9÷48 В. В магазинах не попадались, необходима прямая покупка с AliExpress<sup>2</sup> по почте.

---

<sup>1</sup> БЗП удобен, патрон с ключем дает лучший зажим и возможно точнее

<sup>2</sup> пользуйтесь английской версией — переводная жуткое УГ



Жвигатель Вятка-Автомат 19?? г.



Двигатель печки Камаза



АИРЕ 56 В2, 0.2 КВт



Воздушный шпиндель с цангой ER11

Съемные фрезерные шпиндели, поставляются отдельно или в комплекте с насадкой ручного фрезера по дереву. Лучшие, со стальной шейкой — Kress, активно применяются хобби-ЧПУшниками. Попроще и сильно дешевле делал Интерскол, иногда попадаете понапе. Недостаток как универсального привода — они высокоскоростные, возникают проблемы с понижающими передачами. Применение — приводной высокоскоростной инструмент: боры, фрезы по дереву, микроинструмент для гравиров (микродиски, шарошки). Цанга 8 мм. Для некоторых моделей бывают наборы цанг на мелкий инструмент.



KRESS 530/800/1050 FM(E)  
5600+ р.



Интерскол ФМ-30/750  
/снят с производства/



Интерскол ФМ-55/1000 Э  
5050 р.

## Часть VIII

### Станочное оборудование

Самые распространенные станки — **сверлильные**, т.к. имеют самую простую конструкцию, и минимальную стоимость. Предназначены для самой частой операции: изготовления перпендикулярных круглых дырок в различных материалах, топовые модели имеют также функцию нарезания резьбы. **Для монтажа электроники и кустарного изготовления печатных плат часто используются очень маленькие настольные сверлилки, часто самодельные.**

Наиболее многочисленную группу металлорежущих станков составляют **токарно-винторезные станки**, используются в механических, инструментальных и ремонтных цехах заводов, а также в ремонтных мастерских в основном для обработки деталей, имеющих форму тел вращения. При использовании соответствующей оснастки позволяют растачивать отверстия в призматических (прямоугольных) деталях, и фрезеровать небольшие детали. **Самый ходовой тип детали — тела вращения с наружными и внутренними резьбами: валики, втулки, оси, болты, винты, шпильки, кольца, шайбы и т.д.** К основным размерам, характеризующим токарный станок, относятся

- наибольший допустимый диаметр обрабатываемой заготовки,
- высота **центров** над станиной и
- расстояние между центрами.
- Часто обращают внимание на диаметр **проходного отверстия шпинделя**, определяющий максимальный диаметр **длинномерных заготовок**, что важно при изготовлении партий мелких деталей из длинных прутковых заготовок и нарезке резьб на трубах.

Значительную часть среди металлорежущих станков составляют **фрезерные станки**. Наибольшее распространение имеют консольно-фрезерные. Предназначены для выполнения различных фрезерных работ цилиндрическими, дисковыми, фасонными и другими **фрезами**, можно фрезеровать плоскости, пазы, фасонные поверхности, и т.д. Кроме этого, универсальные консольно-фрезерные станки с поворотным столом или делительной головкой позволяют фрезеровать различного рода винтовые канавки и зубья зубчатых колес. **Для**

самодельной электроники интересны универсальные малогабаритные фрезеры, способные работать в режимах вертикальной и горизонтальной фрезеровки, для изготовления самых разнообразных деталей, а при установке в горизонтальный шпиндель токарной оснастки и небольшую часть токарных работ.

Основными размерами фрезерных станков, по которым можно определить возможность установки и обработки конкретных заготовок с определенными габаритами, являются размеры рабочей поверхности стола (длина и ширина) и **рабочий ход стола/рабочая зона** в продольном, поперечном и вертикальном направлениях. Этими размерами, и типом шпинделя, также определяется возможность установки дополнительного оборудования, выпускаемого серийно: делительных столов, расточных головок, оснастки для нарезки зубчатых колес и т.п.

Общая рекомендация — берите самые большие станки с самой большой рабочей зоной, какие можете себе позволить по цене, помещению для установки, мощностью электропроводки, и стоимостью эксплуатации, обслуживания и расходных материалов. Чем больше станок, тем большую деталь вы сможете изготовить сами, и тем больше возможностей по использованию дополнительного оборудования. Хотя настольные станки дешевы и практически не требуют отдельного помещения, оснастку для них (например поворотный столик) вы для них не найдете, придется ее делать самому или где-то заказывать.



# Глава 12

## Настольные станки

На основе <sup>1</sup>

© Joe Martin, illustration by Craig Libuse

Tabletop Machining

A basic approach to making small parts on miniature machine tools

© Джо Мартин, иллюстрации Craig Libuse

Базовые навыки изготовления мелких деталей на миниатюрных настольных станках

---

<sup>1</sup> копияста: <http://rutracker.org/forum/viewtopic.php?t=3126529>

## Глава 13

# Самодельная оснастка

# Глава 14

## Промышленные станки

Иногда хоббиту удастся получить доступ к старым промышленным станкам. Наиболее богаты в этом плане школы и другие учебные заведения, у них часто где-нибудь в углу или подвале стоят пара старых станков производства СССР. Несмотря на то, что стоят они годами без движения, добраться до них получается с большим гемором: нужно как минимум иметь официальный документ о наличии разряда токаря, фрезеровщика или станочника широкого профиля. Кроме того, без получения какого-то официального статуса, а соответственно и кучи ненужных обязанностей, допуск к станку вы тоже не получите.

Общественных открытых технологических площадок в России не существует в принципе, большая часть станочного оборудования установлена на закрытых территориях, или гниет в подвалах и школах.

## 14.1 1A616: станок токарно-винторезный



- 14.1.1 Назначение и области применения
- 14.1.2 Распаковка и транспортировка
- 14.1.3 Фундамент станка, монтаж и установка
- 14.1.4 Подготовка станка к первоначальному пуску
- 14.1.5 Паспортные данные
- 14.1.6 Описание основных узлов
- 14.1.7 Смазка
- 14.1.8 Первоначальный пуск
- 14.1.9 Указания по технике безопасности
- 14.1.10 Настройка
- 14.1.11 Регулирование
- 14.1.12 Ведомость комплектации

## Часть IX

Разработка ПО для встраиваемых систем

# Глава 15

## IDE

IDE — Integrated Development Environment, интегрированная среда разработки.

Программный пакет, включающий

- средства управления проектом,
- отслеживание зависимостей между файлами (в т.ч. с анализом исходного текста программ на конструкции типа `#include`, `module`, `uses`),
- автозапуском компиляторов для изменившихся файлов,
- GUI для отладчиков (`gdb`),
- специализированный редактор plain text<sup>1</sup> файлов с




---

<sup>1</sup> файлы не включающие непечатаемых символов и бинарных данных, которые можно причитать простым выводом на экран командами типа **type**, **cat**, **more**

- цветовой и шрифтовой **подсветкой синтаксиса**,
  - **автодополнением**: дописываются имена объектов программ, синтаксические конструкции и параметры функций,
  - **автоформатированием**: фрагмент текста переформатируется в соответствии с синтаксисом языка редактируемого файла, проставляются отступы в зависимости от вложенности синтаксических конструкций типа циклов и условных блоков)
  - выделением строк, на которые указывают сообщения об ошибках компиляторов,
  - маркеры точек останова отладчика
- отображение структуры программ, например деревья классов и структур данных
  - контекстные справочники по используемым языкам программирования, автоматический вывод списка параметров при вводе имени функции
  - отображение дизассемблерных листингов для компилируемых языков
  - отображение браузера как вкладки или MDI окна
  - отображение вывода **статических анализаторов** программ с кликабельными ссылками
  - вывод компиляторов и трансляторов с цветовым выделением и переход на ошибочную строку в редакторе при щелчке на ошибке
  - ...

В этой книге рассмотрены три бесплатных мультиплатформенных OpenSource IDE, в порядке навороченности, универсальности, и требуемым ресурсам для работы самой среды:



1.  **ECLIPSE** 15.1: самая навороченная и ресурсоемкая IDE, написана на Java, имеет десятки дополнительных модулей на все случаи, умеет работать со всеми распространенными языками программирования, жрет память, и требует современного компьютера минимум с 2+ Гб ОЗУ. Последний релиз  **ECLIPSE** Luna работает заметно быстрее (особенно при запуске).
2. **Code::Blocks** 15.2: легкая среда для разработки на C/C<sub>+</sub><sup>+</sup>, для других языков может потребоваться написать свои модули или файлы описания синтаксиса
3.  **Vim** 15.3: самый легкий и **портатбельный** универсальный текстовый редактор с расширенными функциями, работает на всех существующих платформах (кроме совсем уж embedded), использует минимум ресурсов, но требует некоторого обучения даже чтобы выйти из vim ☺

## 15.1 ☰eclipse



### 15.1.1 Установка ☰eclipse под ☰Windows

☰ + R >> <https://eclipse.org/> >> Download >> Eclipse Luna release for >> ☰Windows

Качаем архив базовой системы: Eclipse IDE for Java Developers >> ☰Windows 32/64 Bit

Или сразу сборку CDT☰ECLIPSE: Eclipse IDE for C/C++ Developers >> ☰Windows 32/64 Bit

### 15.1.2 Установка ☰eclipse под Linux

☰ + R >> <https://eclipse.org/> >> Download >> Eclipse Luna release for >> Linux

Качаем архив базовой системы: Eclipse IDE for Java Developers >> Linux 32/64 Bit

Или сразу сборку CDT⊕ECLIPSE: Eclipse IDE for C/C++ Developers » Linux 32/64 Bit

Пока качается, параллельно устанавливаем в систему Java-рантайм:

```
sudo aptitude install openjdk-7-jre
```

Распаковывем полученный архив `eclipse-java-luna-SR1-linux-gtk-x86_64.tar.gz` в `$HOME`:

```
cd ~
tar zx < Downloads/eclipse-java-luna-SR1-linux-gtk-x86_64.tar.gz
ls -la eclipse/eclipse
-rwxr-xr-x 1 user user 74675 Авг 13 16:12 eclipse/eclipse
```

Прописываем запуск ⊕ECLIPSE в ваш оконный менеджер или `.blackboxmenu` с параметром `-noSplash` для лечения глюка с запуском на x64-битных системах:

`.blackbox.menu`

### 15.1.3 Установка CDT

**CDT** — расширение ⊕ECLIPSE для разработки на  $C/C_+^+$ , редактирования make-файлов. Это расширение критически важно для вашей работы, поэтому ставить его обязательно, или сразу качать сборку CDT⊕ECLIPSE.



OK  
Work with > CDT  
CDT Main Features > ☒ C/C++ Development Tools  
CDT Optional Features  
Парсер файлов исходников на диалекте C99: ☒ C99 LR Parser  
Поддержка gcc в режиме кросс-компиляции: ☒ GCC Cross Compiler Support  
Аппаратная отладка через gdb: ☒ GDB Hardware Debugging  
Next > Next > Accept > Finish

## 15.1.4 Установка PyDev

PyDev — расширение для разработки на Python:

⊞ ECLIPSE > Help > Install New Software...  
Work with > Add > Add repository  
Name > PyDev  
Location > <http://pydev.org/updates>  
OK  
Work with > PyDev  
PyDev > ☒ PyDev for Eclipse  
Next > Next > Accept > Finish > Certificate > Restart Eclipse > Ok

## 15.1.5 Установка TeXlipse

Если планируете работать с документацией в формате  $\text{\LaTeX}$ , установите расширение TeXlipse:

⊞ ECLIPSE > Help > Install New Software...

Work with >> Add >> Add repository  
Name >> TeXlipse  
Location >> http://texlipse.sourceforge.net/  
OK  
Work with >> TeXlipse

Это расширение поддерживает подсветку синтаксиса, автодополнение, построение динамического оглавления, автокомпиляцию по сохранению, и несколько визардов создания проекта.

## 15.1.6 Редактирование файлов в формате XML и производных

Установите пакет ☯ECLIPSE WST:

Help >> Install New Software  
Work with: >> Luna - http://download.eclipse.org/releases/luna  
Filter: >> WST >> Eclipse WST >> Next >> Next >> Restart >> OK

## 15.1.7 Проверка орфографии

2


То, что проверка орфографии очень удобная вещь вряд ли нужно объяснять. Есть конечно люди, которые не обращают на неё внимание, но это чаще всего из-за экономии времени и отсутствия удобных средств проверки.

Действительно, удобная автоматическая проверка орфографии есть в офисных пакетах, но мне сложно представить разработчика, который будет переносить комментарии в Word и обратно ☺.

Поэтому очень удобно иметь [проверку правописания прямо в IDE](http://www.simplecoding.org/proverka-orfografii-v-eclipse.html). И ☯ECLIPSE в этом смысле полностью соответствует ожиданиям.

---

<sup>2</sup> копияста: <http://www.simplecoding.org/proverka-orfografii-v-eclipse.html>

Долго объяснять что к чему нет смысла. Проверка орфографии встроена в  ECLIPSE и если вы пишете только на английском, то может быть не захотите ничего менять.

Кроме того, есть статья Aaron'a (en) в которой автор рассказывает о подключении дополнительных словарей и плагине **eSpell**.

Но [русских словарей в дистрибутиве нет](#), а при подключении внешних есть нюансы. Поэтому мы максимально подробно рассмотрим [подготовку и добавление русских словарей](#).

Первый вопрос. В каком виде должны быть словари и где их взять?

Тут всё просто. Формат словаря — обычный текстовый файл, в котором каждое слово начинается с новой строки. И нам вполне подойдут свободно распространяемые словари **aSpell**.

Установка состоит из 4 шагов:

1. качаем aSpell и словари для нужных языков

 + R >> <http://aspell.net/win32/>

Binaries >> Full installer

Precompiled dictionaries >> English

Precompiled dictionaries >> Russian

2. устанавливаем сначала **aSpell**, потом отдельно каждый словарь

**Aspell-0.50-3-3-Setup.exe** >> Setup GNU Aspell >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next

Additional >> Next >> Install >> Next >> ☐ View manual >> Finish

**Aspell-en-0.50-2-3.exe** >> Aspell English Dictionary >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next >> Install >> Finish

**Aspell-ru-0.50-2-3.exe** >> Aspell Russian Dictionary >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next >> Install >> Finish

3. делаем дампы словарей, перекодировываем из koi8r в utf8 и объединяем

 + R cmd

```
1 cd \GnuWin32\Aspell
2 bin\aspell dump master en > en.dict
3 bin\aspell dump master ru > ru.koi8
4 iconv -f koi8-r -t utf-8 < ru.koi8 > ru.dict
5 copy en.dict + ru.dict enru.dict
```

4. настраиваем [spell-checker](#)  ECLIPSE

 ECLIPSE >> Window >> Preferences >> Editors >> Text editors >> Spelling

User defined dictionary >> **C:/GnuWin32/Aspell/enru.dict**

Encoding >> UTF-8

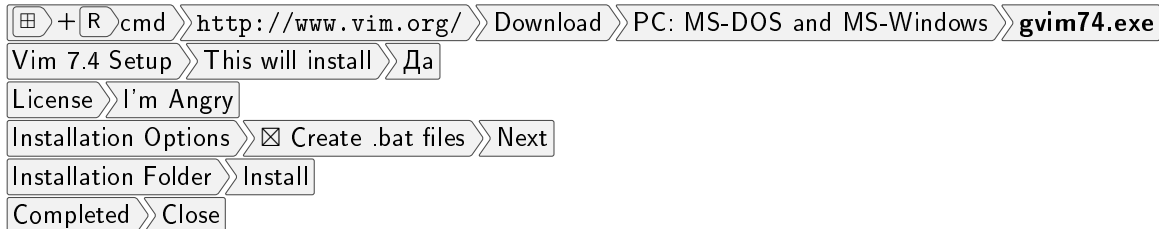
Apply >> OK

## 15.2 Code::Blocks

## 15.3 (g)Vim



### 15.3.1 Установка под Windows





Do you want to see README» **Да**

Теперь можно настроить темную тему и выключение подсветки синтаксиса, по умолчанию после установки используется светлая тема и подсветка выключена:

меню» Правка» Настройка запуска

Переходим в конец файла и включаем **режим вставки**

Ctrl + Down Ins Enter Enter

```
1 syntax on
2 colorscheme pablo
```

Выходим в **режим команд** и принудительно сохраняем



Esc: w ! Enter Enter

**Выходим из (g)Vim**

Esc: q ! Enter

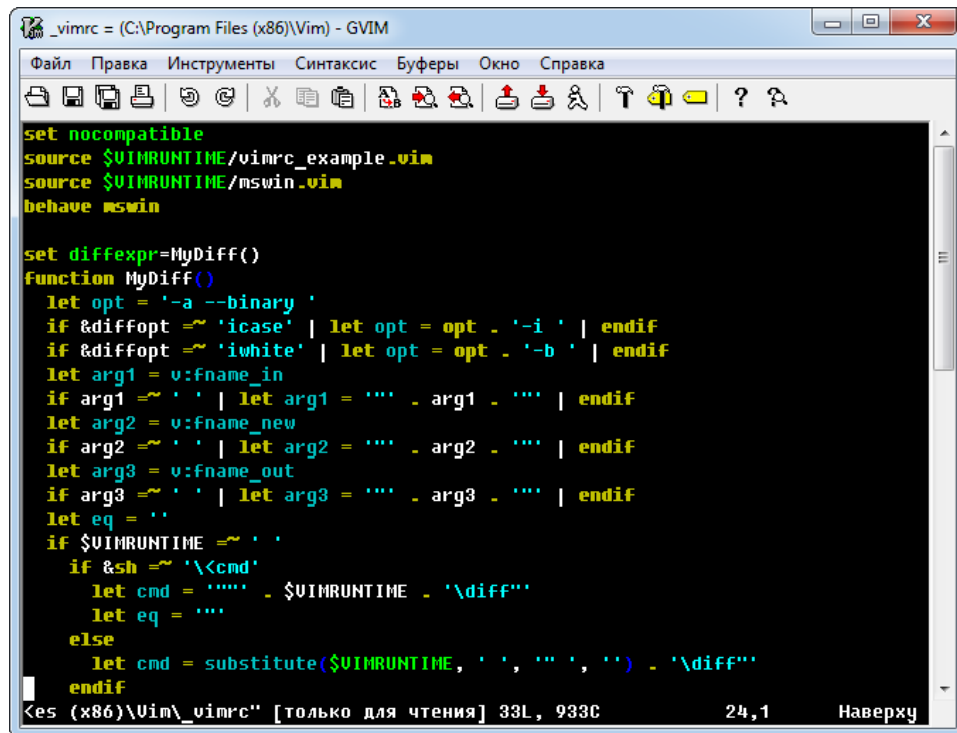
Если не получилось (под Windows 7):

⌘ + R cmd » /Program Files (x86)/Vim/

Копируем файл **\_vimrc** в любой каталог, например в **/tmp/**, затем   Edit with Vim, и повторяем редактирование еще раз.

Затем копируем **\_vimrc** обратно в **/Program Files (x86)/Vim/** с заменой.

Если теперь открыть на редактирование тот же файл, или любой другой текстовый, получим более удобный вид: для файлов известных типов будет работать подсветка синтаксиса.



The screenshot shows the GVIM editor window titled "\_vimrc = (C:\Program Files (x86)\Vim) - GVIM". The menu bar includes "Файл", "Правка", "Инструменты", "Синтаксис", "Буферы", "Окно", and "Справка". The toolbar contains icons for file operations, editing, and navigation. The main text area displays a Vim script with syntax highlighting: keywords like "set", "source", "behave", "function", "let", "if", "endif", and "else" are in green, while comments and other text are in yellow. The script includes settings for "nocompatible", "source" of runtime files, and a custom "MyDiff" function. The status bar at the bottom shows "es (x86)\Vim\\_vimrc" [только для чтения] 33L, 933C, 24,1, and "Наверх".

```
set nocompatible
source $VIMRUNTIME/vimrc_example.vim
source $VIMRUNTIME/mswin.vim
behave mswin

set difffexpr=MyDiff()
function MyDiff()
  let opt = '-a --binary '
  if &diffopt =~ 'icase' | let opt = opt . '-i ' | endif
  if &diffopt =~ 'iwhite' | let opt = opt . '-b ' | endif
  let arg1 = v:fname_in
  if arg1 =~ ' ' | let arg1 = '"' . arg1 . '"' | endif
  let arg2 = v:fname_new
  if arg2 =~ ' ' | let arg2 = '"' . arg2 . '"' | endif
  let arg3 = v:fname_out
  if arg3 =~ ' ' | let arg3 = '"' . arg3 . '"' | endif
  let eq = ''
  if $VIMRUNTIME =~ ' '
    if &sh =~ '\<cmd'
      let cmd = '""' . $VIMRUNTIME . '\diff'
      let eq = ''
    else
      let cmd = substitute($VIMRUNTIME, ' ', '" ', '') . '\diff'
    endif
  endif
endfunction
```

es (x86)\Vim\\_vimrc" [только для чтения] 33L, 933C 24,1 Наверх

## 15.3.2 Выход из (g)Vim

Esc : ! q Enter

## 15.3.3 Выход с автосохранением

`Esc` `Shift` + `Z` `Shift` + `Z`

## 15.3.4 Переход в режим редактирования

(g)Vim запускается в [командном режиме](#), для перехода в режим редактирования используются следующие клавиатурные команды:

- `Ins` или `i`: включение [режима вставки](#) по текущему положению курсора
- `Ins``Ins` или `r`: включение [режима перезаписи](#) поверх текста после курсора
- `Shift` + `A`: включение режима вставки [в конец текущей строки](#)

## 15.3.5 Переход в режим команд

`Esc`

## 15.3.6 Запись редактируемого файла

`Esc` : `w` `Enter`

Если выводится предупреждение типа “файл защищен от записи” или подобное, может сработать принудительная запись:

`Esc` : `!` `w` `Enter`

## 15.3.7 Перезагрузка файла

Для перезагрузки возможно измененного извне файла или отмены всех несохраненных изменений

 :  

## 15.3.8 Отмена последних изменений (undo)

   . . .

## Глава 16

**Make:** управление сборкой проектов

# Глава 17

## VCS: системы контроля версий

### 17.1 CVS

### 17.2 Subversion

### 17.3 Git

#### 17.3.1 GitHub

## Глава 18

# Вспомогательные скрипты на языке Python

# Глава 19

## Основы Си и $C_+^+$

19.1 Установка MinGW (win32)

19.2 Особенности  $C_+^+$  в embedded

19.3 Сборка кросс-компилятора GNU toolchain



# Глава 20

## Лексический и синтаксический анализ

Очень часто в практике возникает необходимость работы с данными в текстовых форматах — **plain text** файлы, в которых в каком-либо формате (на языке разметки, или **DDL: [D]ata [D]efinition [L]anguage**) описаны данные. И от вас требуется реализовать разбор такого файла, выделяя синтаксические структуры и элементы данных, чтобы в дальнейшем после их преобразования например записать текстовый файл в другом формате.

В таком виде хранятся результаты расчетных программ, работающих в пакетном режиме, данные с измерительных систем, поток данных с приемников GPS<sup>1</sup>, очень популярный мета-формат XML со всеми его частными случаями типа HTML, XLIFF??, OpenDocument, тексты программ для станков с ЧПУ,...

С некоторыми хинтами точно так же можно работать и с бинарными файлами, преобразовав их сначала в текстовую форму (в простейшем случае просто сделав hex dump).

В некоторых случаях необходимо написание трансляторов форматов (текстовых) данных, или даже интерпретаторов/компиляторов языков программирования.

Все эти техники с использованием стандартных утилит **flex** и **bison** будут кратко описаны в этой главе.

---

<sup>1</sup> протокол NMEA 0183

Подробнее эти техники рассмотрены в книгах??, особенно стоит отметить талмуд **DragonBook**:

[?] **Книга Дракона**: Ахо, Сети, Ульман Принципы построения компиляторов.

Habr: Компиляция. 1: лексер

Habr: Компиляция. 2: грамматики

Habr: Компиляция. 3: бизон

Habr: Компиляция. 4: игрушечный ЯП

Habr: Компиляция. 5: нисходящий разбор

Habr: Компиляция. 5 $\frac{1}{2}$ : llvm как back-end

Habr: Компиляция. 6: промежуточный код

<http://ds9a.nl/lex-yacc/cvs/lex-yacc-howto.html>

<http://alumni.cs.ucr.edu/~lgao/teaching/flex.html>

[http://www.capsl.udel.edu/courses/cpeg421/2012/slides/Tutorial-Flex\\_Bison.pdf](http://www.capsl.udel.edu/courses/cpeg421/2012/slides/Tutorial-Flex_Bison.pdf)

## 20.1 Лексер и лексический анализ, утилита **flex**

**Лёксер/сканер** — программа или ее часть, которая

1. получает на вход исходные данные в виде сплошного потока одиночных символов,
2. группирует символы согласно набору правил (заданных **регулярными выражениями**) и
3. отдает на выходе символы, уже сгруппированные в **лексёмы** или **токёны**.

Цель лексера — подготовить последовательность лексем для входа другой программы.

В самых простых случаях на лексер можно возложить простые преобразования текста.

**Лексический анализ** — процесс программного разбора входной последовательности символов<sup>2</sup> с целью получения на выходе последовательности групп символов — **токенов**, имеющих собственное смысловое значение<sup>3</sup>. Как правило, лексический анализ производится в соответствии набора правил определённого **формального, искусственного или компьютерного языка**.

**Компьютерный язык**, а точнее его **грамматика**, задаёт определённый набор лексем, которые могут встретиться на входе лексера, и набор правил, по которым их следует группировать.

Традиционно принято организовывать процесс лексического анализа, рассматривая входную последовательность символов как поток одиночных символов. При такой организации **лексер** самостоятельно управляет выборкой отдельных символов из входного потока.

Распознавание лексем с учетом грамматики обычно производится путём их идентификации согласно идентификаторам токенов, определяемых грамматикой языка. При этом любая последовательность символов входного потока (лексэма), которая согласно грамматике не может быть идентифицирована как токен языка, обычно рассматривается как специальный **токен-ошибка**.

Каждый выделенный токен можно представить в виде парной структуры, содержащей

1. идентификатор токена и
2. саму последовательность символов лексемы, выделенной из входного потока<sup>4</sup>.

Рассмотрим обработку текстового файла: разделение текстового фрагмента на абзацы, запись в формате XLIFF для перевода в системе ABBYY SmartCAT, и обратной трансляции из XLIFF в  $\text{\LaTeX}$ -совместимое форматирование.

---

<sup>2</sup> например, такой как исходный код на одном из языков программирования

<sup>3</sup> подобно группировке букв в слово

<sup>4</sup> запись строки, числа и т. д.

Так как задача построения **лексических анализаторов** является стандартной задачей информатики, был разработан типовой инструмент: **генератор лексических анализаторов flex**. Эта программа транслирует описание лексера на своем высококоуровневом языке в фрагмент программы на языке Си/ $C_+$  или самостоятельную программу. Описание лексера прописывается в .lex-файле в формате:

определения, опции, декларации

%%

правила выделения токенов через регулярки

%%

сишный код

**Комментарии** поддерживаются сишные /\* \*/ комментарии.

**Опции** %option

yywrap	отключает вызов лексером функции yywrap() при достижении конца текущего файла
main	включение типовой функции main() вместо заданной пользователем
case-insensitive	регистро-независимый лексический анализ, большие/маленькие буквы не различаются
yylineno	в глобальной си-переменной yylineno доступен номер текущей строки

**Формат опреления** имя определение:

digit [0-9]

number [\+ \-] {0,1}{digit}+ \. {digit}\*

Декларации на Си прописываются в скобках %{ }%

5

flex лексер

```
1 %option noyywrap
2
3 %{
4 #include <iostream>
5 #include <string>
6 using namespace std;
7 #define YYSTYPE string
8 #include "txt2x1iff.tab.h"
9 %}
10
11 %%
12 \n{2,22}          { yylval = ""; return SEP; }
13 \. \ +            { yylval = yytext[0]; return SEP; }
14 \n               { yylval = " "; return CHAR; }
15 \ * \.{4,444} \ [0-9]+\n { yylval = " ..."; return SEP; }
16 .                { yylval = yytext; return CHAR; }
17 %%
```

bison парсер

```
1
2 %{
3 #include <iostream>
4 #include <string>
```

```

5 using namespace std;
6 #define YYSTYPE string
7 #define YYINITDEPTH 0x10000
8 void yyerror(const char *str) { cerr << "\nerror:" << str << "\n\n"; }
9 extern int yylex();
10 %}
11
12 %token CHAR
13 %token SEP
14
15 %%
16 BLOCK: CHARz | CHARz BLOCK ;
17 CHARz: CHAR { cout<<$$; }
18      | SEP { cout<<$$<<"</source></trans-unit>\n\n<trans-unit><source>"; } ;
19 %%
20
21 int main () {
22     cout << "<?xml version=\"1.0\" encoding=\"utf-8\"?>\n";
23     cout << "<xliiff xmlns=\"urn:oasis:names:tc:xliiff:document:1.2\" version=\"1.2\">\n";
24     cout << "<file>\n";
25     cout << "<body>\n\n<trans-unit><source>";
26     int yyp=yyparse();
27     cout << "</source></trans-unit>\n\n</body>\n";
28     cout << "</file>\n";
29     cout << "</xliiff>\n";
30     return yyp;
31 }

```

.txt

```
1 An Introduction to
2 Practical Electronics ,
3 Microcontrollers and
4 Software Design
5
6 2nd edition
7
8 (c) 01 May-2014
9
10 This work is copyright. No one but the author may sell or distribute this material.
11
12 B. Collis
13 www.techideas.co.nz
```

.xliff

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
3 <file>
4 <body>
5
6 <trans-unit><source>An Introduction to Practical Electronics , Microcontrollers and Software Design</source><target></target></trans-unit>
7
8 <trans-unit><source>2nd edition</source><target></target></trans-unit>
9
10 <trans-unit><source>(c) 01 May-2014</source><target></target></trans-unit>
11
12 <trans-unit><source>This work is copyright.</source><target></target></trans-unit>
```

```
13
14<trans-unit><source>No one but the author may sell or distribute this material.</source><
15
16<trans-unit><source>B. Collis www.techideas.co.nz</source></trans-unit>
17
18<trans-unit><source></source></trans-unit>
19
20</body>
21</file>
22</xliff>
```

## 20.2 Генератор синтаксических анализаторов **bison**

**Синтаксический анализ** — процесс анализа последовательности токенов с определением их грамматической структуры. На этом этапе выделяются **синтаксические ошибки**.

## 20.3 Семантический анализ

**Семантический анализ** — процесс выполнения **семантических проверок**: контроль типов, привязка объектов и т.д.



## 20.4 Оптимизация

Выполнение формальных преобразований структур данных, описывающих компилируемую программу, с целью построения более компактного/быстрого машинного кода.

## 20.5 Кодогенерация

Получение реального машинного кода в бинарном представлении, или в виде ассемблерных текстовых файлов.

## 20.6 Транслятор Паскаля

## 20.7 LLVM и разработка собственных компиляторов

Часть X

Микроконтроллеры Cortex-Mx

# Глава 21

## Отладочные платы

21.1 STM32DISCOVERY /Cortex-M3 STM32F103/

21.2 STM32F4DISCOVERY /Cortex-M4 STM32F407/

21.3 Maple Mini /Cortex-M3 STM32F100/

# Часть XI

## Встраиваемый emLinux

Linux для встраиваемых систем<sup>1</sup> — популярный метод быстрого создания комплекса ПО для больших сложных приложений, работающих на достаточно мощном железе, особенно предполагающих интенсивное использование сетевых технологий.

За счет использования уже существующей и очень большой базы исходных текстов ядра, библиотек и программ для Linux, бесплатно доступных в т.ч. и для коммерческих приложений, можно на порядки сократить стоимость разработки собственных программных компонентов, и при этом получить готовую команду бесплатных сторонних разработчиков, уже знакомых с созданием ПО для Linux.

Из недостатков можно отметить:

- Отсутствие полноценной поддержки режима жесткого реального времени;
- Тяжелое ядро;
  - Поддерживаются только мощные семейства процессоров<sup>2</sup>;
  - Значительные требования по объему ОЗУ и общей производительности;
- Дремуемость техспециалистов, контуженных ТурбоПаскалем и Windowsом;

Для сборки emLinux-системы используется метод **кросс-компиляции**, когда используется **кросс-тулчейн**, компилирующий весь комплект ПО для компьютера с другой архитектурой. Типичный пример — сборка ПО на ПК с процессором Intel i7 для Raspberry Pi или планшета на процессоре AllWinner/Tegra/...

emLinux очень широко применяется на рынке мобильных устройств<sup>3</sup>, и устройств интенсивно использующих сетевые протоколы (роутеры, медиacentры).

В качестве примера применения рассмотрим относительно простое приложение: многофункциональные настенные часы с синхронизацией времени через Internet, с будильником, медиапроигрывателем, блэкджеком и плюшками.

---

<sup>1</sup> будем называть его **emLinux**

<sup>2</sup> 32-бит, необходим блок MMU

<sup>3</sup> в т.ч. является основой Android

# Глава 22

## Загрузчик syslinux

Самый простой и удобный загрузчик для i386-систем, ставится на флешку из под Windows, работает с FAT-разделами, поддерживает загрузку с флешек, CDROM и по сети.

<http://www.syslinux.org/>

### 22.1 Закачка

**.zip** с бинарной сборкой **syslinux**:

<https://www.kernel.org/pub/linux/utils/boot/syslinux/syslinux-6.03.zip>

**memtest86+** — полезная утилита для тестирования ОЗУ:

<http://www.memtest.org/download/5.01/memtest86+-5.01.zip>

Если планируете устанавливать рабочую станцию для сборки azLinux с флешки, нужно скачать полные или **netinst** установочные **.iso**-образы:

[Установочный образ Debian Linux i386:](#)

<http://cdimage.debian.org/debian-cd/7.7.0/i386/iso-cd/debian-7.7.0-i386-netinst.iso>

[Установочный образ Debian Linux amd64:](#)

<http://cdimage.debian.org/debian-cd/7.7.0/amd64/iso-cd/debian-7.7.0-amd64-netinst.iso>

[Сборка HDD-инсталлятора i386:](#)

<http://http.us.debian.org/debian/dists/wheezy/main/installer-i386/current/images/hd-media/initrd.gz>

<http://http.us.debian.org/debian/dists/wheezy/main/installer-i386/current/images/hd-media/vmlinuz>

[Сборка HDD-инсталлятора amd64:](#)

<http://http.us.debian.org/debian/dists/wheezy/main/installer-amd64/current/images/hd-media/initrd.gz>

<http://http.us.debian.org/debian/dists/wheezy/main/installer-amd64/current/images/hd-media/vmlinuz>

## 22.2 Установка под ☐Windows на флешку

Распакуйте файлы из .zip

/bios/win32/syslinux.exe

консольный инсталлятор

/bios/com32/menu/menu.c32

модуль текстового меню

/bios/com32/menu/vesamenu.c32

модуль графического меню (VESA)

[служебные библиотеки syslinux](#)

/bios/com32/libutil/libutil.c32

/bios/com32/lib/libcom32.c32

Для установки [syslinux](#) на флешку с FAT, на которую назначена буква F:, выполните батник:

```
1 syslinux -i -m -a -d syslinux F:  
2 pause
```

-i install установить  
-m MBR в MBR  
-a active сделать раздел активным  
-d directory в каталог **syslinux**

Если ставите Debian, распакуйте из **debian-netinst.iso** в **F:/Debian/**

<b>debian-7.7.0-amd64-netinst.iso</b>	.iso-образ установочного CD-ROM
<b>debian-7.7.0-i386-netinst.iso</b>	.iso-образ установочного CD-ROM
<b>hd-media/install.amd/vmlinuz</b>	ядро amd64 (x64)
<b>hd-media/install.amd/initrd.gz</b>	ramdisk с инсталляром
<b>hd-media/install.386/vmlinuz</b>	ядро i386 (x32)
<b>hd-media/install.386/initrd.gz</b>	ramdisk с инсталляром

## 22.3 syslinux.cfg

**syslinux** настраивается текстовым файлом **syslinux.cfg**.

```
1 UI vesamenu.c32  
2 MENU RESOLUTION 640 480  
3 MENU TITLE azLinux  
4 MENU BACKGROUND /syslinux/splash640x480.png
```



```
5
6 DEFAULT azmicro
7 LABEL azmicro
8 MENU LABEL azLinux micro
9 KERNEL /azLinux/micro.kernel
10 INITRD /azLinux/micro.initrd
11 APPEND vga=none
12
13 LABEL azclock
14 MENU LABEL azLinux clock
15 KERNEL /azLinux/clock.kernel
16 INITRD /azLinux/clock.initrd
17 APPEND vga=ask
18
19 LABEL memtest
20 MENU LABEL memtest86+
21 KERNEL /syslinux/memtest.krn
22
23 LABEL debian64
24 MENU LABEL Debian GNU/Linux 7.7.0—amd64—netinst
25 KERNEL /Debian/amd64/vmlinuz
26 INITRD /Debian/amd64/initrd.gz
27 APPEND vga=none
28
29 LABEL debian32
30 MENU LABEL Debian GNU/Linux 7.7.0—i386—netinst
31 KERNEL /Debian/i386/vmlinuz
32 INITRD /Debian/i386/initrd.gz
```

В примере показана реализация с использованием графического VESA меню. Для использования более надежного текстового меню замените на `UI menu.c32`.

Обратите внимание на возможность включения нестандартных видеорежимов используя **[VESA]MENU RESOLUTION**: этот финт нужен для включения графики на ASUS EeePC 701: режим 800×480 недоступен для включения через параметр ядра `vga=`, поэтому приходится использовать возможности `syslinux`.

# Глава 23

## azLinux

Чтобы разобраться как можно собрать встраиваемый Linux, в этом разделе описан набор make-файлов и файлов конфигурации для сборки минимального emLinux. Это обрезанный форк проекта Cross Linux, подробнее описанного в разделе ???. Ограничено количество поддерживаемого железа, упрощены конфигурационные файлы, минимизировано количество библиотек и программных пакетов.

Изначально идея создания этой системы появилась из желания заменить тухлую связку x86/DOS/Turbo-Pascal на что-то

- **более переносимое**: на энергоэффективное ARM/MIPS-железо, в т.ч. (типа)отечественного производства,
- **стабильное**: с полноценной многозадачностью, защитой памяти и данных, и
- **позволяющее использовать максимум возможностей аппаратуры**: большая ОЗУ, ECC, gcc-оптимизированный 32/64-битный код, USB, CAN, Ethernet, WiFi, разнообразные носители данных, аппаратный watchdog.

- Также большой интерес представляют [десятки готовые библиотек](#) сжатия и кодирования данных, численных методов, ЦОС, и обработки изображений, а также
- множество [готовых программ, доступных в исходных кодах](#)<sup>1</sup> для выполнения различных полезных функций: сетевые серверы, символьная математика, обработка данных,...
- Еще одна ключевая фишка — [способность Linux полностью загружаться в ОЗУ с любых носителей, в т.ч. заблокированных на запись](#). Это важно для случаев, когда возможны внезапные выключения питания: вся система работает в ОЗУ-диске, а корректность записи данных на изменяемые носители можно гибко контролировать программно. При запуске после аварийного выключения никаких проверок файловых систем не требуется, ОС стартует сразу, а проверку/починку разделов данных возможно выполнять в фоновом режиме.
- [Время запуска системы](#) — на x86 удалось экспериментально получить время запуска [0.2 сек от загрузки ядра до начала выполнения пользовательского кода](#). Используя модульное ядро, возможно выполнить критический к времени запуска пользовательский код до инициализации USB, сети, внешних носителей данных и тяжелых сервисов.

**Почему не BuildRoot** Эта система сборки создавалась как [максимально облегченный пакет для решения узких задач](#), и для освоения технологии кросс-компиляции. Предполагается что функциональное наполнение не будет развиваться шире набора:

- ядро (реального времени)
- $\mu$ libc
- урезанная командная оболочка (busybox)

---

<sup>1</sup> для использования как есть, изучения принципов работы и модификации под собственные нужды

- несколько прикладных библиотек поддержки (сжатие, кодирование, базовая графика)
- пользовательский узкоспециализированный код на Си/C<sup>+</sup>

Расширять функционал, добавляя libQt, X Window, Apache, MySQL, . . . , Gnome/KDE и т.д. не планируется в принципе — это система для решения узких прикладных задач на аппаратуре с минимальными ресурсами<sup>2</sup>. Интерактивная работа с пользователем также не предполагается, доступна только командная консоль<sup>3</sup>, и очень ограниченные графические и мультимедийные возможности. Если ваши хотелки выходят за этот функционал, рекомендую сразу уходить на использование широко известной системы кросс-сборки Linux-систем под названием BuildRoot??.

## 23.1 Требования к системе сборки (BUILD-хост)

Требования жесткие — 2х-ядерный процессор, 2+ Гб ОЗУ, для 4+ Гб ОЗУ нужен 64х-битный дистрибутив Linux (рекомендую Debian), и естественно никаких виртуалок. Возможна установка системы на флешку, в этом случае требования к ОЗУ еще более ужесточаются — потребуется каталог с временными файлами смонтировать как tmpfs:

добавить в /etc/fstab			
1	tmpfs	/home/user/Azbuka/azlin/tmp	tmpfs auto,uid=user,gid=user 0 0
2	tmpfs	/home/user/Azbuka/azlin/src	tmpfs auto,uid=user,gid=user 0 0
3	tmpfs	/home/user/.ccache	tmpfs auto,uid=user,gid=user 0 0

Можно попытаться сделать билд-сервер и на худшем железе, но будьте готовы к тормозам или внезапному окончанию памяти — ресурсоемка сборка тяжелых библиотек типа libQt или крупных пакетов типа gcc.

<sup>2</sup> особенно интересны процессорные модули в DIMM форм-факторе, только CPU, RAM, NAND и GPIO гребенка

<sup>3</sup> ее можно считать сервисным режимом

Вы можете попробовать поставить Linux на виртуалку, на флешку, и на жесткий диск (если найдете место) и оценить возможности этих вариантов на сборке пакета `gcc0`. При сборке с флешки на ноутбуке с 2 Гб ОЗУ мне для сборки `gcc0` пришлось временно размонтировать `cross/src`, сделать `./mk.rc && make gcc ramclean`, а потом примонтировать `tmpfs` опять на `src`.

Сборка под MinGW/Cygwin совершенно неживая. Если совсем никак без винды — используйте виртуалки, и будьте готовы ждать.

## 23.2 Понятие пакет

Прежде чем продолжить, введем понятие `пакет`. В azLinux `пакетом` называется одна или несколько частей скриптов сборки, обозначаемых именем. В чем-то это похоже на бинарные пакеты обычных дистрибутивов Linux — чтобы добавить в систему какой-то функционал, мы устанавливаем `бинарный пакет`. Но есть и отличие: пакет дистрибутива это реальный архивный файл, содержащий в себе файлы программ, данных; в azLinux пакет — виртуальная штука с именем.

Просматривая файлы в каталоге `mk/`, легко найти имена пакетов по шаблону:

```
.PHONY: somename
somename: [зависимые файлы]
    [команда1]
    ...
```

Если вы запустите команду:

```
cd ~/az ; ./mk.rc && make somename
```

запустится `сборка пакета somename`.

Но не нужно забывать, что кроме этой секции в `.mk`, существуют зависимости между файлами, при работе команд сборки динамически создаются и изменяются файлы, иногда что-то скачивается из Interneta — все эти процессы тоже входят в пакет.

Часть пакетов не связана со сборкой программ, а выполняют служебные функции, поэтому для них правильнее будет фраза **запуск пакета**.

Все действия выполняются с помощью команды **make**. Обратите особое внимание на то, что **Makefile** собирается скриптом **mk.rc** из частей в каталоге **mk/**, поэтому **если вы что-то меняете в скриптах, не забудьте сначала запустить ./mk.rc**.

mk.rc

```
1 #!/ bin / sh
2 cat \
3     mk/head.mk \
4     mk/dirs.mk \
5     mk/versions.mk \
6     mk/packages.mk \
7     mk/commands.mk \
8     mk/clean.mk \
9     mk/gz.mk \
10    mk/src.mk \
11    mk/cfg.mk \
12    mk/cross.mk \
13    mk/core.mk \
14    mk/kernel.mk \
15    mk/ulibc.mk \
16    mk/busybox.mk \
17    mk/user.mk \
```

```
18 mk/libs.mk \
19 mk/apps.mk \
20 mk/emu.mk \
21 mk/root.mk \
22 mk/boot.mk \
23 > Makefile
```

## 23.3 Клонирование проекта azLinux

При необходимости вносить правки<sup>4</sup> работайте с вашим собственным форком на GitHub. Получите клон пакета из репозитория:

```
cd ~ ; git clone --depth=1 -o gh https://github.com/user/azlin az
```

При необходимости обновитесь:

```
cd ~/az ; git pull
```

## 23.4 Общий порядок сборки

Каждый пакет собирается командой:

```
./mk.rc && make [HW=rpi] [APP=clock] <package>
```

### 1. dirs создание дерева каталогов 23.6

---

<sup>4</sup> что естественно — вам потребуется добавлять свои пакеты и поддержку железа



2. **gz** закачка архивов исходников 23.7
3. **tc** сборка кросс-компилятора 23.14.6
  - (a) **binutils** ассемблер, линкер и утилиты 23.14.7
  - (b) **cclibs** библиотеки для сборки **gcc** 23.14.8
  - (c) **gcc0** сборка минимального кросс-компилятора Си 23.14.9
4. **core** сборка основной системы 23.14.11
  - (a) **kernel** ядро Linux 23.14.12
  - (b) **ulibc** библиотека **uClibc** 23.14.13
  - (c) **busybox** набор утилит **busybox** 23.14.15
  - (d) **gcc** пересборка полного кросс-компилятора Си/ $C_{++}$  23.14.14
5. **libs** сборка библиотек **\${LIBS}** 23.14.16
6. **apps** сборка прикладных пакетов **\${APPS}** 23.14.17
7. **user** сборка пользовательского кода 23.14.18
8. **root** формирование корневой файловой системы 23.14.19
9. **boot** сборка загрузчика 23.14.20 **syslinux/grub/uboot**
10. **emu** запуск собранной системы в эмуляторе 23.14.24
11. **netboot** сетевая загрузка 23.15
12. **firmware** прошивка на устройство 23.16

## 23.5 Фиксация переменных

Если вам требуется собрать систему со значениями переменных, отличающихся от тех, которые прописаны в make-файлах, при запуске **всех** пакетов нужно указывать требуемые значения в командной строке. Это позволит легко выбрать нужный вам вариант сборки.

```
./mk.rc && make HW=rpi APP=clock distclean dirs tc core libs apps boot root
```

При таком указании все переназначения для этих переменных игнорируются, поэтому возможны некоторые сложности с указанием например опций оптимизации.

## 23.6 **dirs**: Создание дерева каталогов

После загрузки или обновления запустите пакет **dirs**:

```
$ ./mk.rc && make dirs
mkdir -p /home/user/Azbuka/azlin/gz /home/user/Azbuka/azlin/src
/home/user/Azbuka/azlin/tmp /home/user/Azbuka/azlin/x86_64-linux-gnu
/home/user/Azbuka/azlin/qemu386-clock /home/user/Azbuka/azlin/qemu386-clock/boot
```

```
$ ls -la
итого 72
-rwxr-xr-x 1 user user 121 Дек 8 11:43 mk.rc
drwxr-xr-x 2 user user 4096 Дек 8 11:43 mk
drwxr-xr-x 2 user user 4096 Дек 8 11:43 app
drwxr-xr-x 2 user user 4096 Дек 8 11:43 hw
drwxr-xr-x 2 user user 4096 Дек 8 11:43 cpu
```

```
drwxr-xr-x 2 user user 4096 Дек 8 11:43 arch
drwxr-xr-x 2 user user 4096 Дек 8 13:26 gz
drwxr-xr-x 2 user user 4096 Дек 8 11:43 app
drwxr-xr-x 3 user user 4096 Дек 8 13:26 qemu386micro
drwxr-xr-x 2 user user 4096 Дек 8 13:26 qemu386micro.cross
-rw-r--r-- 1 user user 147 Дек 8 11:43 README.md
-rw-r--r-- 1 user user 17699 Дек 8 13:25 azlin.tex
drwxr-xr-x 2 user user 4096 Дек 8 13:26 src
drwxr-xr-x 2 user user 4096 Дек 8 13:26 tmp
-rw-r--r-- 1 user user 2087 Дек 8 13:26 Makefile
```

Пакет **dirs** прописан в файле

mk/dirs.mk

```
1 # directories processing
2 # sw sources mirror archive in .tar.[GZ]
3 GZ = $(PWD)/gz
4 # [S]ou[RC]e codes unpacked
5 SRC = $(PWD)/src
6 # [T]e[MP] build dirs
7 TMP = $(PWD)/tmp
8 # build/target triplets
9 BUILD = $(shell gcc -dumpmachine)
10 # target root filesystem
11 ROOT = $(PWD)/$(HW)$(APP)
12 # cross-compiler [T]ool[C]hain
13 TC = $(ROOT).cross
14 BOOT = $(ROOT)/boot
```

```
15 ETC = $(ROOT)/etc
16 USR = $(ROOT)/usr
17 USRBIN = $(USR)/bin
18 USRLIB = $(USR)/lib
19 DIRS = $(GZ) $(SRC) $(TMP) $(TC) $(ROOT) $(BOOT) $(USR) $(USRBIN) $(USRLIB)
20 .PHONY: dirs
21 dirs:
22 ____mkdir -p $(DIRS)
```

Встроенная переменная **PWD** содержит полное имя каталога, из которого был запущен **make**.

Каталог зеркала архивов исходных текстов программ

GZ

```
1 GZ = $(PWD)/gz
```

Каталог распаковки исходных текстов: некоторые пакеты должны собираться в дереве исходников

SRC

```
1 SRC = $(PWD)/src
```

Каталог out-of-tree сборки: остальные пакеты умеют собираться вне дерева исходников, если хватает ОЗУ  
этот каталог удобно мониторить как **tmpfs**

TMP

```
1 TMP = $(PWD)/tmp
```

Переменная **BUILD** задает **триплет** системы, на которой вы собираете: x86\_64-linux-gnu, i686-linux-gnu или что-то подобное. Для получения триплета используется подстановка строки, выдаваемой запуском **gcc**.

## BUILD

```
1 BUILD = $(shell gcc -dumpmachine)
```

Каталог в который собирается кросс-компилятор. Используется триплет рабочей Linux-системы.

## TC

```
1 TC = $(ROOT).cross  
2 TC = $(ROOT)/etc
```

Каталог целевой **rootfs**. Отдельно прописан загрузочный каталог, в который будет записываться собранное ядро, образ **initrd**, бинарники и конфиги загрузчика. Имя **ROOT** создается из двух переменных, описанных далее: имя аппаратной платформы **HW23.9** и имени приложения **APP23.8**.

## ROOT/BOOT

```
1 ROOT = $(PWD)/$(HW)$(APP)  
2 BOOT = $(ROOT)/boot
```

Список всех рабочих каталогов в одной переменной:

## DIRS

```
1 DIRS = $(GZ) $(SRC) $(TMP) $(TC) $(ROOT) $(BOOT) $(USR) $(USRBIN) $(USRLIB)
```

## 23.7 gz: Загрузка архивов исходников

### mk/gz.mk

```
1 # download sw packages sources to gz/  
2
```

```
3 .PHONY: gz
4 gz:
5 ____make gz_cc
6 ____make gz_core
7 ____make gz_libs
8
9 .PHONY: gz_cc
10 gz_cc:
11 ____$(WGET) http://ftp.gnu.org/gnu/binutils/$(BINUTILS).tar.bz2
12 ____$(WGET) http://gcc.skazkaforyou.com/releases/$(GCC)/$(GCC).tar.bz2
13 ____$(WGET) ftp://ftp.gmplib.org/pub/gmp/$(GMP).tar.bz2
14 ____$(WGET) http://www.mpfr.org/mpfr-current/$(MPFR).tar.bz2
15 ____$(WGET) http://www.multiprecision.org/mpc/download/$(MPC).tar.gz
16
17 .PHONY: gz_core
18 gz_core:
19 ____$(WGET) https://www.kernel.org/pub/linux/kernel/v3.x/$(KERNEL).tar.xz
20 ____$(WGET) http://www.uclibc.org/downloads/$(ULIBC).tar.xz
21 ____$(WGET) http://busybox.net/downloads/$(BUSYBOX).tar.bz2
22
23 .PHONY: gz_libs
24 gz_libs:
25 ____$(WGET) https://www.libsdl.org/release/$(SDL).tar.gz
26 ____$(WGET) https://www.libsdl.org/projects/SDL_image/release/$(SDL_IMAGE).tar.gz
27 ____$(WGET) http://download.sourceforge.net/libpng/$(PNG).tar.xz
28 ____$(WGET) http://zlib.net/$(ZLIB).tar.xz
```

## 23.8 APP: Приложение

**Приложение** — короткое кодовое название вашего варианта сборки системы в целом.

Приложение задается в файле **mk/head.mk** через переменную **APP**, доступные значения:

1. **micro**: минимальная версия системы, только командная консоль
2. **clock**: простые Linux-powered электронные часы

Значение переменной **APP** по умолчанию задано в **mk/head.mk**:

APP @ hw/head.mk

```
1 APP = micro
```

Если вам нужно собрать другое приложение, вы можете переопределить значение из командной строки при запуске **vsx** пакетов:

```
./mk.rc && make APP=micro distclean dirs tc core libs apps
```

В **app/\${APP}.mk** в переменных задаются:

- **LIBS**: набор используемых библиотек
- **PACKS**: набор используемых программных пакетов 23.14

app/micro.mk

```
1 # app: micro
2 LIBS =
3 APPS = hello
```

```
1 # app: clock
2 LIBS = zlib sdl
3 APPS = hello
```

## 23.9 HW: Поддерживаемое железо

Конфигурация целевого железа задается в файле `mk/head.mk` через переменную `HW`, доступные значения приведены в таблице:

HW	CPU	ARCH	RAM	HD	SD	USB	Eth	WiFi	GPIO
qemu386	i486sx	i386	32M+	<input type="checkbox"/> IDE		<input type="checkbox"/>	ne2k		
eeepc701	CeleronM	i386	512M+	<input type="checkbox"/> SSD	<input type="checkbox"/> SD	<input checked="" type="checkbox"/>	A??	<input type="checkbox"/> AR2425	
gac1037	Celeron1037U	x86_64	1G+	<input type="checkbox"/> SATA		<input checked="" type="checkbox"/>	2×RTL8111		
qemuARM		arm	32M+						
cubie1	AllWinnerA10	armhf	1G		<input type="checkbox"/> μSD	<input checked="" type="checkbox"/>			
rpi	BCM2835	armel	512M		<input type="checkbox"/> SD	<input checked="" type="checkbox"/>			
qemuMIPS		mips	32M+						
mr3020	AR7240	mips	32M	4M		<input checked="" type="checkbox"/>		<input type="checkbox"/> AR9331	
vocore	RT5350	mips	32M	8M		<input type="checkbox"/>		<input type="checkbox"/> SoC	
bswift		AR9331	mips	64M	16M	<input type="checkbox"/>			20+

```
1 # [H]ard [W]are: qemu386 qemuARM qemuMIPS cubie1 rpi ...
2 HW = qemu386
3 # [APP]lication: micro clock
4 APP = micro
```



```
5 # load extra hw definitions: ARCH CPU ...
6 include hw/$(HW).mk
7 # load extra defs for CPU setted in $(HW).mk
8 include cpu/$(CPU).mk
9 # load extra defs for ARCHitecture setted in $(CPU).mk
10 include arch/$(ARCH).mk
11 # load extra app defs: LIBS APPS ...
12 include app/$(APP).mk
```

## 23.10 i386

Персоналки с архитектурой **i386** — самое сложное семейство с точки зрения поддержки. Комбинации процессоров, материнских плат и плат расширений дают сотни вариантов конфигураций, в т.ч. десятки моделей компьютеров в формате PC/104 и промышленных панелей. Особенно доставляет тот факт, что 95% периферии имеет закрытые бинарные драйвера только для Windows, поэтому будьте аккуратны с выбором железа.

### 23.10.1 qemu386: эмулятор QEMU

hw/qemu386.mk

```
1 # QEMU emulator: i386 mode
2 CPU = i486sx
3 QEMU_CFG = -m 32M -net none
```

### 23.10.2 eeepc701: ASUS Eee PC 701

1 # ASUS Eee PC 701

2 CPU = CeleronM

### 23.10.3 gac1037: Gigabyte GA-C1037UN-EU rev.2



CPU	Celeron 1037U
ОЗУ	2 × DDR3, max 16G, 2 канала
чипсет	Intel NM70
сеть	2 × Realtek® GbE 1Gb (rtl8111)
HDD	2 × SATA2 (3Gb/s), 1 × SATA3 (6Gb/s), 1 × eSATA
USB	6 × USB3.0
видео	IntelGMA, выходы на VGA D-Sub и HDMI 1.4
аудио	Realtek ALC887 (HDA)
PCI	×1

В качестве варианта 64-битной платформы взята портативная материнка для неттопов: Gigabyte GA-C1037UN-EU. На ней возможны два варианта сборки:

1. [x86\\_64/x64](#): нативный CPU=Celeron1037u

2. [i686/x32](#): режим совместимости со старыми процессорами CPU=i686

<http://www.ixbt.com/news/hard/index.shtml?17/33/31>

На текущий момент эта материнка — оптимальный вариант для офисного, рабочего неигрового компьютера, или базы для изготовления мобильной рабочей станции в миникейсе: комплект из GA-C1037UN-EU и блока питания стоит порядка 5 тыс.руб, при этом возможна установка до 2×8G ОЗУ, что пока недоступно на дешевых ноутбуках. Но — **отвратительный радиатор на мосте NM70, нагрев до 70°C, в обязательном порядке делать сквозную продувку корпуса до включения материнки.**

материнская плата	GA-C1037UN-EU rev.2
CPU	Celeron 1037U (впаян)
охлаждение	отвратительное, обязательны кулера на все чипы и сквозная продувка корпуса,
ОЗУ	1×8G DDR3 (в планах 2×8G)
HDD	нет, используется флешка 8G USB3 Transcend JF750 (возможно SATA SSD)
TFT	китайский 3.5” автомониторчик + конвертер HDMI2RCA на случай посмотреть видеовывод, обычно везде где я использую эту поделку, есть VGA монитор или хотя бы большой телевизор
электропитание	БП Hipro HPE-350W + автоинвертор батарея не требуется, под боком всегда есть 220 или 12 В по необходимости в транспорт грузится пара заряженных автоаккумуляторов

hw/gac1037.mk

```
1 # Gigabyte GA-C1037UN-EU
2 CPU = Celeron1037U
```

## 23.11 ARM

### 23.11.1 qemuARM: эмулятор QEMU

**23.11.2 cubie1: Cubie Board v.1**

**23.11.3 rpi: Raspberry Pi model B**

**23.12 MIPS**

**23.12.1 qemuMIPS: эмулятор QEMU**

**23.12.2 mr3020: роутер MR3020**

mr3020

**23.12.3 vocore: VoCore**

vocore

**23.12.4 bswift: BlackSwift**

bswift

## 23.13 CPU: Конфигурации процессоров

Настройки на процессор задаются в файле `cpu/${CPU}.mk`.

<b>ARCH</b>	архитектура целевой системы, используется при конфигурировании ядра
<b>TARGET</b>	триплет целевой системы, параметр задает тип целевой системы при сборке кросс-компилятора и используется во всех скриптах <code>configure</code> при сборке остальных пакетов
<b>CFG_CPU</b>	параметры при сборке кросс-компилятора
<b>CPU_FLAGS</b>	параметры <code>gcc</code> для оптимизации кода

### 23.13.1 i386

`cpu/i486sx.mk`

```
1 # i486sx cpu
2 # target arch is Intel x86 (32 bit)
3 ARCH = i386
4 # target triplet for embedded linux
5 TARGET = i486-linux-uclibc
6 # target CPU configure params
7 CFG_CPU = --disable-mmx --disable-3dnow --disable-sse
```

`cpu/CeleronM.mk`

```
1 # Intel Celeron M ULV 353
2 ARCH = i386
3 TARGET = celeronm-linux-uclibc
```

`cpu/Celeron1037U.mk`

```
1 # Intel Celeron 1037U
2 ARCH = i386
3 TARGET = celeronm-linux-uclibc
```

## 23.13.2 ARM

## 23.13.3 MIPS

cpu/AR7240.mk

```
1 # Atheros AR7240 CPU (400Mhz)
2 ARCH = mips
3 TARGET = mips-linux-uclibc
```

cpu/RT5350.mk

```
1 # Ralink RT5350 360MHz
2 ARCH = mips
3 TARGET = mips-linux-uclibc
```

## 23.14 Пакеты

### 23.14.1 versions.mk: Версии пакетов

### 23.14.2 tc: кросс-компилятор

mk/versions.mk

```
1 BINUTILS_VER = 2.24
2 GMP_VER = 5.1.3
3 MPFR_VER = 3.1.2
4 MPC_VER = 1.0.2
5 GCC_VER = 4.9.1
```

## 23.14.3 **core:** ядро

mk/versions.mk

```
1 KERNEL_VER = 3.17.6
2 ULIBC_VER = 0.9.33.2
3 BUSYBOX_VER = 1.22.1
```

## 23.14.4 **boot:** загрузчики

mk/versions.mk

## 23.14.5 **libs:** библиотеки

mk/versions.mk

```
1 SDL_VER = 1.2.15
2 SDL_IMAGE_VER = 1.2.12
```



## 23.14.6 tc: сборка кросс-компилятора

```
1
2 CFG_BINUTILS = --target=$(TARGET) $(CFG_ARCH) $(CFG_CPU) \
3 ____--with-sysroot=$(ROOT) \
4 ____--with-native-system-header-dir=/include \
5 ____--enable-lto
6
7 CFG_CCLIBS = --disable-shared \
8 ____--with-gmp=$(TC) --with-mpfr=$(TC) --with-mpc=$(TC)
9
10 CFG_GMP = $(CFG_CCLIBS)
11 CFG_MPFR = $(CFG_CCLIBS)
12 CFG_MPC = $(CFG_CCLIBS)
13
14 CFG_GCC0 = $(CFG_BINUTILS) $(CFG_CCLIBS) \
15 ____--disable-shared --disable-threads \
16 ____--without-headers --with-newlib \
17 ____--enable-languages="c"
18
19 CFG_GCC = $(CFG_BINUTILS) $(CFG_CCLIBS) \
20 ____--enable-threads --enable-libgomp \
21 ____--enable-languages="c,c++"
22
23 .PHONY: tc
24 tc: binutils cclibs gcc0
25
```

```
26 .PHONY: binutils
27 binutils: $(SRC)/$(BINUTILS)/README
28 ____rm -rf $(TMP)/$(BINUTILS) && mkdir $(TMP)/$(BINUTILS) &&\
29 ____cd $(TMP)/$(BINUTILS) &&\
30 ____$(SRC)/$(BINUTILS)/$(BCFG) $(CFG_BINUTILS) &&\
31 ____$(MAKE) && make install-strip
32
33 .PHONY: cclibs
34 cclibs: gmp mpfr mpc
35
36 .PHONY: gmp
37 gmp: $(SRC)/$(GMP)/README
38 ____rm -rf $(TMP)/$(GMP) && mkdir $(TMP)/$(GMP) &&\
39 ____cd $(TMP)/$(GMP) &&\
40 ____$(SRC)/$(GMP)/$(BCFG) $(CFG_GMP) &&\
41 ____$(MAKE) && make install-strip
42
43 .PHONY: mpfr
44 mpfr: $(SRC)/$(MPFR)/README
45 ____rm -rf $(TMP)/$(MPFR) && mkdir $(TMP)/$(MPFR) &&\
46 ____cd $(TMP)/$(MPFR) &&\
47 ____$(SRC)/$(MPFR)/$(BCFG) $(CFG_MPFR) &&\
48 ____$(MAKE) && make install-strip
49
50 .PHONY: mpc
51 mpc: $(SRC)/$(MPC)/README
52 ____rm -rf $(TMP)/$(MPC) && mkdir $(TMP)/$(MPC) &&\
53 ____cd $(TMP)/$(MPC) &&\
```

```
54 ____$(SRC)/$(MPC)/$(BCFG) $(CFG_MPC) &&\
55 ____$(MAKE) && make install-strip
56
57 .PHONY: gcc0
58 gcc0: $(SRC)/$(GCC)/README
59 ____rm -rf $(TMP)/$(GCC) && mkdir $(TMP)/$(GCC) &&\
60 ____cd $(TMP)/$(GCC) &&\
61 ____$(SRC)/$(GCC)/$(BCFG) $(CFG_GCC0)
62 ____cd $(TMP)/$(GCC) && $(MAKE) all-gcc
63 ____cd $(TMP)/$(GCC) && $(MAKE) install-gcc
64 ____cd $(TMP)/$(GCC) && $(MAKE) all-target-libgcc
65 ____cd $(TMP)/$(GCC) && $(MAKE) install-target-libgcc
66
67 .PHONY: gcc
68 gcc: $(SRC)/$(GCC)/README
69 ____rm -rf $(TMP)/$(GCC) && mkdir $(TMP)/$(GCC) &&\
70 ____cd $(TMP)/$(GCC) &&\
71 ____$(SRC)/$(GCC)/$(BCFG) $(CFG_GCC)
72 ____cd $(TMP)/$(GCC) && $(MAKE) all-gcc
73 ____cd $(TMP)/$(GCC) && $(MAKE) install-gcc
74 ____cd $(TMP)/$(GCC) && $(MAKE) all-target-libgcc
75 ____cd $(TMP)/$(GCC) && $(MAKE) install-target-libgcc
```

## 23.14.7 **binutils**: ассемблер, линкер и утилиты

<code>-target=\$(TARGET)</code>	триплет целевой платформы
<code>\$(CFG_ARCH)</code>	параметры архитектуры
<code>\$(CFG_CPU)</code>	параметры процессора
<code>-program-prefix</code>	префикс <code>&lt;prefix&gt;-(as ld ..) </code>
<code>-with-sysroot</code>	каталог в котором находятся хедеры и библиотеки
<code>-with-native-system-header-dir=/include</code>	каталог с хедерами
<code>-enable-lto</code>	<code>[L]ink[T]ime [O]ptimization</code>

arch/i386.mk

```
1 CFG_ARCH = --disable-multilib
```

—disable-multilib    выключить смешанный 32/64-битный режим

arch/arm.mk

```
1 CFG_ARCH = --enable-interwork
```

—enable-interwork    разрешить смешанный код ARM/THUMB

cpu/i486sx.mk

```
1 # i486sx cpu
2 # target arch is Intel x86 (32 bit)
3 ARCH = i386
4 # target triplet for embedded linux
5 TARGET = i486-linux-uclibc
6 # target CPU configure params
7 CFG_CPU = --disable-mmx --disable-3dnow --disable-sse
```

## 23.14.8 cclibs: библиотеки для сборки gcc

gmp mpfr mpc

## 23.14.9 gcc0: сборка минимального кросс-компилятора Си

При сборке gcc0/gcc отключайте ccache: кэш при разовых сборках не работает, но при монтировании как tmpfs потребляет слишком много ОЗУ:

```
./mk.rc && make CCACHE= gcc0
```

## 23.14.10 gcc: пересборка полного кросс-компилятора Си/C<sub>+</sub>

Пакет собирается после сборки core.

## 23.14.11 core: сборка основной системы

mk/core.mk

```
1 .PHONY: core
2 core: kernel ulibc busybox gcc
```

## 23.14.12 kernel: ядро Linux

mk/kernel.mk

```
1 CFG_KERNEL = ARCH=$(ARCH) INSTALL_HDR_PATH=$(ROOT)
2
3 .PHONY: kernel
4 kernel: $(SRC)/$(KERNEL)/README
5 ____# 1
6 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) distclean
7 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) allnoconfig
8 ____# 2
9 ____cat kernel/all >> $(SRC)/$(KERNEL)/.config
10 ____cat kernel/arch/$(ARCH) >> $(SRC)/$(KERNEL)/.config
11 ____cat kernel/cpu/$(CPU) >> $(SRC)/$(KERNEL)/.config
12 ____cat kernel/hw/$(HW) >> $(SRC)/$(KERNEL)/.config
13 ____cat kernel/app/$(APP) >> $(SRC)/$(KERNEL)/.config
```

```

14 ____# 3
15 ____echo "CONFIG_CROSS_COMPILE=\"$(TARGET)-\" >> $(SRC)/$(KERNEL)/.config
16 ____echo "CONFIG_LOCALVERSION=\"-$(HW)$(APP)\" >> $(SRC)/$(KERNEL)/.config
17 ____echo "CONFIG_DEFAULT_HOSTNAME=\"$(HW)$(APP)\" >> $(SRC)/$(KERNEL)/.config
18 ____# 4
19 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) menuconfig
20 ____# 5
21 ____cd $(SRC)/$(KERNEL) && $(MAKE) $(CFG_KERNEL)
22 ____# 6
23 ____make kernel-$(ARCH)-fix
24 ____cp $(SRC)/$(KERNEL)/arch/x86/boot/zImage $(BOOT)/$(HW)$(APP).kernel
25 ____# 7
26 ____cd $(SRC)/$(KERNEL) && make $(CFG_KERNEL) headers_install
27
28 .PHONY: kernel-i386-fix
29 kernel-i386-fix:
30 ____cp $(SRC)/$(KERNEL)/arch/x86/boot/bzImage \
31 ____$(SRC)/$(KERNEL)/arch/x86/boot/zImage

```

**ARCH** архитектура: `src/linux-x.x.x/arch/*`

**INSTALL\_HDR\_PATH** путь установки **хедеров ядра**

1. подготовка к сборке с пустым **конфигом** `src/linux-x.x.x/.config`
2. накатываем на пустой **.config** файлы-модификаторы, содержащие переопределения переменных конфигурации:
  - **all** универсальные параметры ядра для всех платформ
  - **arch** параметры, специфичные для архитектуры

- **cpu** параметры, адаптирующие ядро к конкретному процессору (эмулятор FPU, возможности управления тактовой частотой и потреблением, поддержка многоядерности и т.п.)
- **hw** параметры чипсета и периферии материнской платы (**модули ядра** = драйвера)
- **app** параметры, в основном **отключаемые** для конкретного **приложения** (отключение графики, режим реального времени, спец.параметры)

3. установка параметров кросс-компиляции и имени сборки
4. запуск интерактивного меню конфигурирования, выйти с сохранением
5. сборка ядра
6. копирование готового файла ядра в **\${BOOT}**
7. генерация **хедеров** в **\${ROOT}**

Все параметры ядра идут с префиксом **CONFIG\_**:

- **core.mk**
  - **CROSS\_COMPILE** префикс кросс-компилятора `<px>-(gcc|as|ld..)`
  - **LOCALVERSION** суффикс ядра `linux-x.x.x-suffix`
  - **DEFAULT\_HOSTNAME** имя компьютера `<host>.<domain>`
- **all для всех платформ**
  - **NOHIGHMEM=y** сборка в режиме  $\leq 1\text{ Gb}$  ОЗУ



- `KERNEL_GZIP=y` ядро сжимать алгоритмом `gzip`<sup>5</sup>
- **корневая файловая система в `initrd/ramdisk`**
  - \* `BLK_DEV_INITRD=y` rootfs в O3Y(`initrd`)
  - \* `PROC_FS=y` файловая подсистема `/proc`
  - \* `SYSFS=y` файловая подсистема `/sys`
  - \* `DEVTMPFS=y` файловая подсистема `/dev` (интерфейсы драйверов устройств)
  - \* `DEVTMPFS_MOUNT=y` автоматически монтировать `devfs`
- **режим реального времени**<sup>6</sup> ??
  - \* `PREEMPT=y` **вытесняющая многозадачность** в режиме ядра
  - \* `HZ_1000=y` системный таймер 1 KHz
- **исполняемые форматы файлов**
  - \* `BINFMT_ELF=y` бинарные файлы в формате `ELF`
  - \* `BINFMT_SCRIPT=y` файлы скриптов с `#!/bin/sh` в заголовке
  - \* `COREDUMP=n` не писать **корки** при сбоях
- **поддержка консоли `tty`**
  - \* `TTY=y` последовательные порты и командная консоль
  - \* `UNIX98_PTYS=n` псевдотерминалы UNIX98
  - \* `LEGACY_PTYS=n` псевдотерминалы UNIX/BSD
- **клавиатура**
  - \* `INPUT_KEYBOARD=y` ввод с аппаратной клавиатуры

---

<sup>5</sup> важно для загрузчиков на не-i386 системах

<sup>6</sup> повышенная или **гарантированная** отзывчивость системы на внешние события

– **мышь**

- \* **INPUT\_MOUSE=y**
- \* **INPUT\_MOUSEDEV=y**
- \* **INPUT\_MOUSEDEV\_PSAUX=n** не создавать /dev/psaux

– **USB HID: устройства ввода без драйверов (клавиатура, мышь, джойстик, самодельные кнопки)**  
включение поддержки USB на i386 не требуется

- \* **HID=y** [H]uman [I]nterface [D]evice
- \* **HID\_GENERIC=y** универсальный драйвер USB HID Class

– **Графический режим FrameBuffer**

- \* **FB=y**
- \* **FRAMEBUFFER\_CONSOLE=y** командная консоль
- \* **LOGO=y** вывод пингвина при запуске
- \* **LOGO\_LINUX\_MONO=y** выводить только черно-белый вариант
- \* **LOGO\_LINUX\_VGA16=n**
- \* **LOGO\_LINUX\_CLUT224=n**

– **Отладка и printk лог ядра**

- \* **DEBUG\_KERNEL=y** включение отладочных функций ядра
- \* **EARLY\_PRINTK=y** вывод пусковой части ядра
- \* **PRINTK=y** вывод главного системного лога ядра **printk**
- \* **PRINTK\_TIME=y** выводить метки времени (для отладки времени запуска)
- \* **SCHED\_DEBUG=n** не отлаживать планировщик
- \* **DEBUG\_PREEMPT=n** не отлаживать вытесняющий режим

- arch i386

- X86\_GENERIC=y универсальная оптимизация для всех i386-процессоров
- UART/COM/RS232 последовательные порты
  - \* SERIAL\_8250=y UART 8250/16550
  - \* SERIAL\_8250\_CONSOLE=y командная консоль на COM-портах
- VGA\_CONSOLE=y командная консоль на VGA 80×25
- FrameBuffer
  - \* FB\_VESA=y универсальный видеодрайвер VESA 2.0+
- клавиатура
  - \* KEYBOARD\_ATKBD=y клавиатура PC AT / PS2
- мышь
  - \* MOUSE\_PS2=y мышь PS/2
  - \* MOUSE\_SERIAL=y (старая) мышь на RS232
- NVRAM=y доступ к памяти CMOS
- отладка
  - \* MAGIC\_SYSRQ=y волшебная кнопка Alt + SysRq
  - \* X86\_VERBOSE\_BOOTUP=y доп.сообщения при пуске ядра

- cpu i386 i486sx

- M486=y
- MATH\_EMULATION=y включить программную эмуляцию мат.сопроцессора (FPU) в ядре

- **hw** qemu386 qemu386

- **MATH\_EMULATION=n** выключить эмуляцию FPU: **QEMU** запускается на процессорах Pentium и старше, которые всегда имеют аппаратный модуль плавающей точки
- **HZ\_1000=n** в режиме эмуляции быстрый таймер не имеет смысла,
- **HZ\_100=y** переключаемся на 100 Hz

- **app** micro

- **FB=n** выключаем поддержку FrameBuffer,..
- **INPUT\_MOUSE=n** мыши

## 23.14.13 **ulibc**: библиотека **uClibc**

mk/ulibc.mk

```
1 CFG_ULIBC = CROSS=$(TARGET) ARCH=$(ARCH) PREFIX=$(ROOT)
2 #___UCLIBC_EXTRA_CFLAGS=""
3
4 .PHONY: ulibc
5 ulibc: $(SRC)/$(ULIBC)/README
6 ___# 1
7 ___cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) distclean
8 ___cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) allnoconfig
9 ___# 2
10 ___cat ulibc/all >> $(SRC)/$(ULIBC)/.config
11 ___cat ulibc/arch/$(ARCH) >> $(SRC)/$(ULIBC)/.config
```

```

12 ____cat ulibc/cpu/$(CPU) >> $(SRC)/$(ULIBC)/.config
13 ____cat ulibc/app/$(APP) >> $(SRC)/$(ULIBC)/.config
14 ____# 3
15 ____echo "CROSS_COMPILER_PREFIX=\"$(TARGET)-\" >> $(SRC)/$(ULIBC)/.config
16 ____echo "KERNEL_HEADERS=\"$(ROOT)/include\" >> $(SRC)/$(ULIBC)/.config
17 ____# 4
18 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) menuconfig
19 ____# 5
20 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC)
21 ____# 6
22 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) install
23 ____# 7
24 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) install_utils
25 ____# 8
26 ____cd $(SRC)/$(ULIBC) && $(MAKE) $(CFG_ULIBC) hostutils
27 ____cp $(SRC)/$(ULIBC)/utils/ldd.host $(TC)/bin/$(TARGET)-ldd
28 ____cp $(SRC)/$(ULIBC)/utils/ldconfig.host $(TC)/bin/$(TARGET)-ldconfig
29 ____cp $(SRC)/$(ULIBC)/utils/getconf.host $(TC)/bin/$(TARGET)-getconf
30 ____# 9 (in root package)
31 ____# $(LDCONFIG) -v -r $(ROOT)

```

1. подготовка к сборке с пустым **конфигом** `src/uClibc-x.x.x/.config`
2. накатываем на пустой `.config` файлы-модификаторы, содержащие переопределения переменных конфигурации:
  - **all** универсальные параметры `ulibc` для всех платформ
  - **arch** параметры, специфичные для архитектуры

- **сру** параметры, адаптирующие **ulibc** к конкретному процессору (набор команд и оптимизация)
- **app** параметры, в основном **отключаемые** для конкретного **приложения** (отключение shared библиотек и неиспользуемых групп функций)

3. установка параметров кросс-компиляции и **пути к хедерам ядра**

4. запуск интерактивного меню конфигурирования, выйти с сохранением

5. сборка **ulibc**

6. инсталляция в **ROOT**

7. инсталляция утилит для работы с shared библиотеками **на целевой системе**

8. инсталляция утилит для **BUILD-системы**

9. создание **/etc/ld.so.cache** (выполняется в пакете **root**)

- - **ARCH\_HAS\_MMU=y** **всегда включено**
  - **ARCH\_USE\_MMU=y** **всегда включено**
  - **UCLIBC\_HAS\_FPU=y** **всегда включено**: используется эмулятор в ядре
  - **UCLIBC\_HAS\_FLOATS=y** функции плавающей точки
  - **DO\_C99\_MATH=n** функции float math C99
  - **UCLIBC\_CTOR\_DTOR=y** конструктор/деструктор
  - **UCLIBC\_HAS\_LFS=y** большие файлы

- разделяемые **shared** библиотеки

- `HAVE_SHARED=y`
- `LDSO_CACHE_SUPPORT=y` /etc/ld.so.conf
- `LDSO_PRELOAD_ENV_SUPPORT=n` переменная `LD_PRELOAD` содержит список библиотек, загружаемых первыми
- `LDSO_LD_LIBRARY_PATH=n` отключить переменную `LD_LIBRARY_PATH` со списком каталогов поиска **shared** библиотек

- многопоточные **threads** программы

- `LINUXTHREADS_NEW=y` новый вариант потоков

- параметры установки

- `DOSTRIP=y`
- `RUNTIME_PREFIX=`
- `DEVEL_PREFIX=`

- опции необходимые **busybox**

- `UCLIBC_HAS_CTYPE_TABLES=y`
- `UCLIBC_HAS_NETWORK_SUPPORT=y`
- `UCLIBC_HAS_FNMATCH=y`
- `UCLIBC_HAS_GNU_GETOPT=y`
- `UCLIBC_HAS_REGEX=y`

– UCLIBC\_SUSV3\_LEGACY=y

•

ulibc/all

```
1 ARCH_HAS_MMU=y
2 ARCH_USE_MMU=y
3 UCLIBC_HAS_FPU=y
4 UCLIBC_HAS_FLOATS=y
5 #DO_C99_MATH=y
6
7 HAVE_SHARED=y
8 LDSO_CACHE_SUPPORT=y
9 LDSO_PRELOAD_ENV_SUPPORT=n
10 LDSO_LD_LIBRARY_PATH=n
11
12 LINUXTHREADS_NEW=y
13 UCLIBC_CTOR_DTOR=y
14
15 UCLIBC_LINUX_SPECIFIC=y
16 UCLIBC_LINUX_MODULE_26=n
17 UCLIBC_HAS_LFS=y
18
19 UCLIBC_HAS_LFS=y
20 UCLIBC_HAS_REALTIME=y
21 UCLIBC_HAS_STRING_GENERIC_OPT=y
22 UCLIBC_HAS_STRING_ARCH_OPT=y
23
```



```
24 DOSTRIP=y
25 RUNTIME_PREFIX=""
26 DEVEL_PREFIX=""
27
28 # BB needs
29 UCLIBC_HAS_CTYPE_TABLES=y
30 UCLIBC_HAS_NETWORK_SUPPORT=y
31 UCLIBC_HAS_FNMATCH=y
32 UCLIBC_HAS_GNU_GETOPT=y
33 UCLIBC_HAS_REGEX=y
34 UCLIBC_SUSV3_LEGACY=y
35
36 ## Python needs
37
38 #DO_C99_MATH=y
39 #UCLIBC_HAS_LONG_DOUBLE_MATH=n
40 #UCLIBC_HAS_WCHAR=y
41 #UCLIBC_SUSV4_LEGACY=y
42 #UCLIBC_HAS_PTY=y
43 #UNIX98PTY_ONLY=n
44 #UCLIBC_HAS_LIBUTIL=y
45
46 #DOMULTI=y
47 #UCLIBC_HAS_IPV4=y
48 #UCLIBC_HAS_COMPAT_RES_STATE=n
49 #UCLIBC_HAS_SYSLOG=y
50 ## ALSA needs
51 #UCLIBC_HAS_BSD_ERR=y
```

ulibc/arch/i386

```
1 TARGET_i386=y
```

ulibc/cpu/i486sx

```
1 CONFIG_486=y
```

## 23.14.14 gcc: пересборка полного gcc

После сборки **ulibc** необходимо еще раз пересобрать **gcc** с полными настройками.

## 23.14.15 busybox: набор утилит busybox

mk/busybox.mk

```
1 CFG_BUSYBOX = \  
2 ____CONFIG_PREFIX=$(ROOT) \  
3 ____CROSS_COMPILE=$(TARGET)- \  
4 ____SYSROOT=$(ROOT)  
5  
6 .PHONY: busybox  
7 busybox: $(SRC)/$(BUSYBOX)/README  
8 ____# 1  
9 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) distclean  
10 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) allnoconfig  
11 ____# 2  
12 ____cp app/$(APP).bb $(SRC)/$(BUSYBOX)/.config  
13 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) menuconfig
```

```
14 ____cp $(SRC)/$(BUSYBOX)/.config app/${APP}.bb
15 ____# 3
16 ____cd $(SRC)/$(BUSYBOX) && $(MAKE) $(CFG_BUSYBOX) install
```

1. подготовка к сборке с пустым конфигом
2. интерактивное меню конфигурирования с сохранением конфига в `app/${APP}.bb`
3. сборка и инсталляция в **ROOT**

Особенность **busybox** — скрипты конфигурации не умеют обрабатывать переписывание переменных конфигурации, поэтому приходится держать в `app/*.bb` настройки для каждого приложения полностью. Если вы создаете свое приложение, скопируйте `app/micro.bb` в качестве базовой версии вашего `app/userapp.bb`.

**23.14.16** **libs:** сборка библиотек `${LIBS}`

**23.14.17** **apps:** сборка прикладных пакетов `${APPS}`

**23.14.18** **user:** сборка пользовательского кода

**23.14.19** **root:** формирование корневой файловой системы

mk/root.mk

```
1 ROOTREX = ". / ( boot | include | lib / .* \ . [ao] ) "
```

```
2
```

```
3 .PHONY: root
```

```
4 root:
```

```

5 ____# 1
6 ____rm -rf $(ETC) ; cp -r etc $(ROOT)/
7 ____cp README.md $(ETC)/
8 ____chmod +x $(ETC)/init.d/*
9 ____# 2
10 ____$(LDCONFIG) -v -r $(ROOT)
11 ____# 3
12 ____ln -fs /sbin/init $(ROOT)/init
13 ____# 4
14 ____cp -r share $(ROOT)/
15 ____# 5
16 ____cd $(ROOT) && find . | egrep -v $(ROOTREX) | \
17 ____cpio -o -H newc > $(BOOT)/$(HW)$(APP).cpio
18 ____cat $(BOOT)/$(HW)$(APP).cpio | gzip -9 > $(BOOT)/$(HW)$(APP).rootfs

```

1. пересоздание **/etc/**
2. генерация **кеша динамических библиотек** **/etc/ld.so.cache**
3. **/sbin/init** → **/init**
4. **/share/**
5. создание **initrd** используя фильтрацию файлов **регулярным выражением** в переменной **ROOTREX**

**/etc/inittab**

```

1 :: sysinit:/etc/init.d/rcS
2 :: shutdown:/etc/init.d/rcD

```

```
3 :: ctrlaltdel:/sbin/reboot
4 tty1::askfirst:/bin/sh
5 tty2::askfirst:/bin/sh
6 tty3::askfirst:/bin/sh
7 tty4::askfirst:/bin/sh
8 #ttyS1::respawn:/bin/sh
```

/etc/init.d/rcS

```
1 #!/bin/sh
2 # system dirs
3 mkdir /tmp
4 mkdir /proc ; mount -t proc proc /proc
5 mkdir /sys ; mount -t sysfs sysfs /sys
6 # hotplug device manager (automount, firmware boot, config)
7 mdev -s && echo /sbin/mdev > /proc/sys/kernel/hotplug
8 # about
9 uname -a
10 cat /etc/README.md
```

1. создание системных каталогов

2. запуск демона **mdev**, обслуживающего **devfs**: создание/удаление устройств в **/dev**, автомонтирование, загрузка прошивок,...

3. вывод **README**

/etc/init.d/rcD

```
1 #!/bin/sh
2 sync
3 umount -a
```

## 23.14.20 **boot**: сборка загрузчика **syslinux/grub/uboot**

mk/boot.mk

### 23.14.21 **syslinux**

### 23.14.22 **grub**

### 23.14.23 **uboot**

## 23.14.24 **emu**: запуск собранной системы в эмуляторе

Для отладки кода, не связанного жестко с железом, для которого допускается выполнение в эмуляторе, используется **QEMU**.

mk/emu.mk

```
1 include app/$(APP).qemu
2
3 .PHONY: emu
4 emu: $(BOOT)/$(HW)$(APP).kernel $(BOOT)/$(HW)$(APP).rootfs
5 ____qemu-system-$(ARCH) $(QEMU_CFG) \
```

```
6 ____-kernel $(BOOT)/$(HW)$(APP).kernel \
7 ____-initrd $(BOOT)/$(HW)$(APP).rootfs \
8 ____-append "$ (QEMU_VGA)"
```

hw/qemu386.mk

```
1 # QEMU emulator: i386 mode
2 CPU = i486sx
3 QEMU_CFG = -m 32M -net none
```

Переменная QEMU\_CFG задает параметры запуска QEMU:

-m	объем ОЗУ на эмулируемой системе
-net none	отключить сеть и iPXE boot
-kernel	прямая загрузка ядра Linux из файла
-initrd	образ корневой файловой системы (initrd)
-append	параметры ядра:
vga=ask	запрос списка видеорежимов при загрузке и выбор режима
vga=none	std. VGA text 80×25, FrameBuffer отключен
vga=0x315	VESA 800×600×24bit
vga=0x317	VESA 1024×768×16bit

- Mode 0x0301: 640x480 (+640), 8 bits
- Mode 0x0303: 800x600 (+800), 8 bits
- Mode 0x0305: 1024x768 (+1024), 8 bits
- Mode 0x0311: 640x480 (+1280), 16 bits
- Mode 0x0312: 640x480 (+2560), 24 bits
- Mode 0x0314: 800x600 (+1600), 16 bits
- Mode 0x0315: 800x600 (+3200), 24 bits

Mode 0x0317: 1024x768 (+2048), 16 bits

Mode 0x0318: 1024x768 (+4096), 24 bits

23.15 **netboot**: Сетевая загрузка

23.16 Прошивка на устройство

23.17 RT-патч



Глава 24

BuildRoot

## Глава 25

# Особенности OpenWrt

## Часть XII

### Символьная и численная математика

В практике любого инженера математика занимают главную роль. Без хорошего знания математики, причем практически всех областей, от школьной до дифференциального исчисления, работать в этой области практически невозможно.

Прежде всего свободное знание математики, физики, и химии необходимо для чтения любой литературы, если вам нужно разобраться в какой-либо прикладной области. Очень часто приходится реализовывать некоторые численные методы вычислений, выполняющиеся в вашем устройстве в реальном времени, для управления процессами, обработки сигналов с датчиков, принятия решений о включении исполнительных устройств и т.п. Ну и конечно вы не сможете создать само устройство, не понимая принципы его работы ☺. Это конечно не относится к различным простейшим устройствам типа таймеров или простой автоматики, но стоимость заказов такого типа  $\rightarrow 0$ .

Если вы хотите поднять или восстановить свой уровень знания базовых наук (а заодно и английского), удобно воспользоваться ресурсом <https://www.khanacademy.org/>: это знаменитая on-line академия **Khan Academy**, имеющая как набор видеолекций по базовым техническим наукам, так и большую батарею тестов для проверки ваших знаний. Не забывайте периодически проходить все тесты, чтобы поддерживать свои знания рабочими. Из недостатков — отвратнейшая реализация на мобильных устройствах, часть тестов просто не работает, а ввод ответов крайне неудобен из-за необходимости постоянно пользоваться (полно)экранной клавиатурой и переключения на числовой ввод.

На русском языке ресурсов такого класса к сожалению пока не попадалось. Кое-что есть кусочками, но по большей части только лекции в стиле «книжкой по башке», похоже на навыки **вводного** обучения в России просто не существует. Если есть силы и желание, можете сами реализовать проект по созданию онлайн системы базового образования ☺.

В этом разделе собраны примеры проектов, требующие некоторых базовых знаний, а также рассмотрено использование OpenSource программ для вычислений и обработки данных.

# Глава 26

## Общие сведения о компьютерной математике

12

Для начала пару слов о том, что из себя представляют эти самые символьные или, как их еще называют, аналитические вычисления, в отличие от численных расчетов. Компьютеры, как известно, оперируют с числами, целыми и с плавающей запятой<sup>3</sup>. К примеру, решения уравнения  $x^2 = 2x + 1$  можно получить как -0.41421356 и 2.41421356, а  $3x = 1$  — как 0.33333333. А ведь хотелось бы увидеть не приближенную цифровую запись, а точную величину, т. е.  $1 \pm \sqrt{2}$  в первом случае и  $1/3$  во втором. С этого простейшего примера и начинается разница между численными и символьными вычислениями.

---

<sup>1</sup> копия: <http://maxima.sourceforge.net/ru/maxima-tarnavsky-1.html>

<sup>2</sup> Тихон Тарнавский. Maxima — максимум свободы символьных вычислений

<sup>3</sup> на самом деле настоящую “плавучку” поддерживают только достаточно мощные процессоры, не хуже i486dx, встраиваемые не-DSP CPU/MCU аппаратно работают только с целыми числами:  $\pm 127$ ,  $\pm 2^{16-1}$  и  $\pm 2^{32-1}$  в зависимости от разрядности ядра 8- 16- или 32-бит

Но кроме этого, есть еще задачи, которые вообще невозможно решить численно или наоборот аналитически.

Например, параметрические уравнения, где в виде решения нужно выразить неизвестное через параметр; или нахождение производной от функции; да практически любую достаточно общую задачу можно решить только в символьном виде.

Наоборот, для многих задач не существует точного аналитического решения, и приходится применять **численные методы** их решения.

В некоторых случаях нужно получение простого и быстрого **приближенного решения** — это может понадобиться в системах управления, когда микроконтроллер не успевает за управляемым процессом, если пытается получить точное численное решение. При обработке сигналов например не требуется точное решение, достаточно результата, получаемого численными методами.

Для решения аналитических задач давно появились компьютерные программы, оперирующие любыми математическими объектами, от чисел любого типа, векторов и матриц до тензоров, от функций до интегродифференциальных уравнений и т. д. — они имеют общее название [C]omputer [A]lgebra [S]ystem.

Среди математического ПО для аналитических (символьных) вычислений наиболее широко известны коммерческие CAS-пакеты Maple, Mathematica и MathCAD. Для символьных вычислений предназначен пакет MatLab. Это очень мощные и очень дорогие инструменты для ученых и инженеров, позволяющие автоматизировать наиболее рутинную и требующую повышенного внимания часть работы, оперируя при этом аналитической записью данных, т. е. почти математическими формулами.

Такие программы можно назвать средой программирования, с той разницей, что в качестве элементов языка программирования выступают привычные человеку математические обозначения.

Для преподавателей, аспирантов, и студентов предоставляются академические более дешевые лицензии, но для хоббитов и коммерческого применения требуется покупка полной лицензии, имеющей зачастую космическую стоимость. Неплохим вариантом может послужить использование бесплатного и свободного OpenSource программного обеспечения, описанного далее — пакетов **Maxima** и **Octave**.

С другой стороны, основное направление, кроме научных разработок, где такие программы востребованы — высшее образование; а использование для учебных нужд именно свободного ПО — реальная возможность

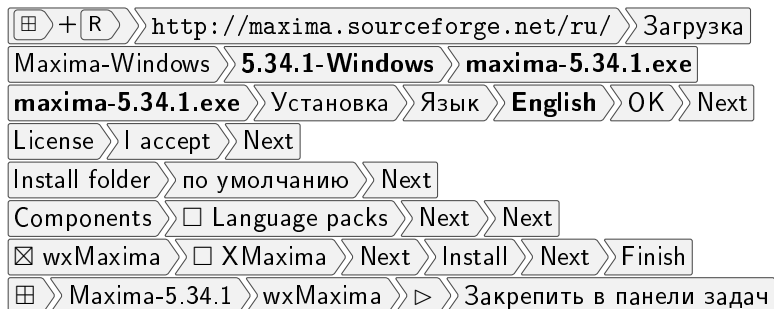
и для ВУЗа, и для студентов и преподавателей иметь в своем распоряжении легальные копии такого ПО без больших, и даже каких-либо денежных затрат.

# Глава 27

## Пакет Maxima

Maxima – пакет CAS: символьной математики, но также включает функционал численных вычислений и визуализации.

### 27.1 Установка Maxima под Windows





Дополнительная документация: <http://maxima.sourceforge.net/ru/documentation.html>

PDF для книги Ильина В.А., Силаев П.К. Система аналитических вычислений Maxima для физиков-теоретиков [?] получена из файла .ps с помощью сервиса <http://ps2pdf.com/convert.htm>.

## 27.2 Калькулятор

## Часть XIII

Подготовка технической документации

# Глава 28

## Верстка в $\text{\LaTeX}$

## Глава 29

# Оформление листингов

# Глава 30

## Подготовка иллюстраций

### 30.1 GIMP

### 30.2 Inkscape

### 30.3 Graphviz

Замечания для соавторов “Абзуки ARMатурщика”

Замечания для соавторов “Абзуки ARMатурщика”

## Часть XIV

### Примерные учебные планы

Глава 31

Блондинко

# Глава 32

## Школотрон

1. Введение в практическую электронику: Вводная электронная схема

постараться понять разделы

Ток и напряжение, сечение проводника, плотность тока

Проводники и изоляторы, сопротивление и проводимость

Диэлектрическая и тепловая прочность изоляции

Масштабные множители

Текст может оказаться сложноватым, но эти понятия необходимы для понимания электроники, как минимум нужно понимать что такое ток, напряжение, сопротивление и мощность.

2. Введение в практическую электронику: Использование мультиметра

Умение хотя бы немного пользоваться простейшим измерительным прибором необходимо: проверить батарейку, посмотреть ток через элемент, определить ноги светодиода и т.п.



# Глава 33

## Студень

1. Введение в практическую электронику

## Глава 34

# Технический специалист

1. Введение в практическую электронику прочитать по диагонали

# Часть XV

Куча

В этот раздел собраны все материалы, не вошедшие в основную часть потому что слишком сложны для начинающих, не попадают не в один раздел по тематике, или не вписались по каким-то другим параметрам.

Все новые материалы также сначала попадают сюда, а потом принимается решение об их переносе в основную часть.

Часто сюда пишут статьи те, кто принимает участие в создании книги эпизодически, или те, у кого нет достаточно времени заниматься их подготовкой.

# Предметный указатель

## программы

memtest86+, 140

syslinux, 140

## термины

Ампер, 26

БМП, 23

ГМ, 25

УЗО, 30

Вольт, 26

автомат, 30

диэлектрическая прочность, 29

диэлектрик, 28

диод, 25

допустимый рабочий ток, 24

электрический ток, 25

элемент, 23

гидравлическая модель, 25

грамматика, 129

источник питания, 25

изолятор, 28

компонент, 23

короткое замыкание, 30

кросс-компиляция, 139

кросс-тулчейн, 139

лексема, 128

лексер, 128

лексический анализ, 129

лексический анализатор, 130

макетная плата, 23

масштабные множители, 31

мощность, 31

напряжение, 25, 26

номинал, 35

общий провод, 23, 26

пакет, 148

площадь сечения проводника, 26

плотность тока, 26

предохранитель, 30

проводимость, 27  
проводник, 25, 28  
разность потенциалов, 26  
регулярное выражение, 128  
резистор, 24, 25, 28  
сборка, 139  
сборка пакета, 148  
сечение проводника, 26  
семантический анализ, 134  
сила тока, 26  
синтаксический анализ, 134  
сканер, 128  
сопротивление, 27, 29  
светодиод, 24  
шина питания, 23  
ток, 25, 26  
ток короткого замыкания, 30  
токен, 128  
учебный план, 12  
устройство защитного отключения, 30  
вафля, 23  
выгорание проводки, 30  
вывод элемента, 28  
выводной корпус, 35  
закон Джоуля Ленца, 31  
закон Ома  
    для постоянного тока, 28

замкнутый контур, 24  
земля, 23  
breadboard, 23  
Data Definition Language, 127  
DDL, 127  
emLinux, 139  
plain text, 127

## azLinux

настройка  
    приложение, 157  
пакет  
    dirs, 154  
переменная  
    APP, 157  
    BOOT, 155  
    BUILD, 155  
    DIRS, 155  
    GZ, 154  
    ROOT, 155  
    SRC, 154  
    TC, 155  
    TMP, 154  
приложение  
    clock, 158  
    micro, 158  
скрипт

mk.rc, 150  
железо, 158  
AR7240, 165  
ASUS Eee PC 701, 160  
BlackSwift, 163  
Celeron1037U, 165  
CeleronM, 164  
Gigabyte GA-C1037UN-EU, 162  
i386, 159  
i486sx, 164  
QEMU, 159, 163  
RT5350, 165  
VoCore, 163